AUTOMATIC 3D FACIAL PERFORMANCE ACQUISITION AND ANIMATION USING MONOCULAR VIDEOS

A Dissertation

by

FUHAO SHI

Submitted to the Office of Graduate and Professional Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Chair of Committee,	Jinxiang Chai
Committee Members,	John Keyser
	Scott Schaefer
	Ergun Akleman
Head of Department,	Dilma Da Silva

May 2017

Major Subject: Computer Science

ABSTRACT

Facial performance capture and animation is an essential component of many applications such as movies, video games, and virtual environments. Video-based facial performance capture is particularly appealing as it offers the lowest cost and the potential use of legacy sources and uncontrolled videos. However, it is also challenging because of complex facial movements at different scales, ambiguity caused by the loss of depth information, and a lack of discernible features on most facial regions. Unknown lighting conditions and camera parameters further complicate the problem.

This dissertation explores the video-based 3D facial performance capture systems that use a single video camera, overcome the challenges aforementioned, and produce accurate and robust reconstruction results.

We first develop a novel automatic facial feature detection/tracking algorithm that accurately locates important facial features across the entire video sequence, which are then used for 3D pose and facial shape reconstruction. The key idea is to combine the respective powers of local detection, spatial priors for facial feature locations, Active Appearance Models (AAMs), and temporal coherence for facial feature detection. The algorithm runs in realtime and is robust to large pose and expression variations and occlusions.

We then present an automatic high-fidelity facial performance capture system that works on monocular videos. It uses the detected facial features along with multilinear facial models to reconstruct 3D head poses and large-scale facial deformation, and uses per-pixel shading cues to add fine-scale surface details such as emerging or disappearing wrinkles and folds. We iterate the reconstruction procedure on large-scale facial geometry and fine-scale facial details to improve the accuracy of facial reconstruction. We further improve the accuracy and efficiency of the large-scale facial performance capture by in-

troducing a local binary feature based 2D feature regression and a convolutional neural network based pose and expression regression, and complement it with an efficient 3D eye gaze tracker to achieve realtime 3D eye gaze animation. We have tested our systems on various monocular videos, demonstrating the accuracy and robustness under a variety of uncontrolled lighting conditions and overcoming significant shape differences across individuals.

ACKNOWLEDGMENTS

First of all, I would like to express my deep gratitude to my advisor Prof. Jinxiang Chai for his continuous guidance and help throughout my PhD study. During the past few years, I have learned a lot from him on analyzing problems, dealing with details, writing and presentation skills, and the determination to get things done. Without his inspiration, this work would not have been possible, and I greatly appreciate his expertise and invaluable advice during the projects. I would also like to thank my committee members, Prof. John Keyser, Prof. Scott Schaefer, and Prof. Ergun Akleman, for their precious comments and encouragements.

I would also like to thank all my friends and colleagues in our department, Peizhao Zhang, Jianjie Zhang, Peihong Guo, Yen-lin Chen, Jianyuan Min, Xiaolin Wei, Donghui Han, Josiah Manson, Jason Smith, Billy Clack, and Yan Lu, for the help, ideas, and joy they brought into my work and life.

My special thanks to Dr. Xin Tong and Hsiang-Tao Wu at Microsoft Research Asia and Congyi Wang at the Institute of Computing Technology (Chinese Academy of Sciences) for their collaborations and invaluable discussions on my facial performance capture projects.

Last but not the least, I would like to take the opportunity to thank my wife, Tiana Zhang. No matter good times or bad, she is always the one supporting me, devoting her love and time. Thank you for being in my life, and it is so beautiful with you.

CONTRIBUTORS AND FUNDING SOURCES

Contributors

This work was supported by a dissertation committee consisting of Professor Jinxiang Chai, Professor John Keyser, and Professor Scott Schaefer of the Department of Computer Science and Engineering and Professor Ergun Akleman of the Department of Visualization.

The results of applications in Section 3 were produced by Hsiang-Tao Wu at Microsoft Research Asia. He also helped render the results. The eye gaze tracking algorithm and corresponding experiments and evaluations in Section 4 were conducted by Congyi Wang at Institute of Computing Technology (Chinese Academy of Sciences). The results of applications in Section 4 were conducted jointly by Congyi Wang and the student.

All other work conducted for the dissertation was completed by the student, under the advisement of Professor Jinxiang Chai of the Department of Computer Science and Engineering.

Funding Sources

This work is partially supported by the National Science Foundation under Grants No. IIS-1055046 and IIS-1065384, and National Natural Science Foundation of China under Grants No.61173055 and No.60970086.

Its contents are solely the responsibility of the authors and do not necessarily represent the official views of the National Science Foundation.

TABLE OF CONTENTS

Page

LIST OF FIGURES

LIST OF TABLES

TABLE Page

1. INTRODUCTION

Facial animation is an essential component of many applications such as movies, video games, and virtual environments. Thus far, one of the most popular and successful approaches for creating virtual faces often involves capturing facial performances of real people. The facial performance of the actor/user, which includes the large-scale deformation, fine-scale facial details such as wrinkles and folds, and other indispensable components such as eye movement, are captured and transferred to virtual characters. A famous example is the facial expression production in the movie "Avatar", in which the actors wear markers on the face and control the virtual characters in realtime. However, the whole system is invasive, very expensive, and requires professionality to operate. Recently, low-cost facial performance capture systems that can be used for common users are becoming popular, especially in the hot area of virtual/augmented reality. For example, Faceshift uses depth sensor such as Kinect to capture the 3D facial expression and enables realtime communication through virtual avatars. Still, the requirement for depth sensor limits its applications to everyday life such as the usage on mobile devices. This motivates us to explore algorithms and systems on the video-based facial performance reconstruction, which is particularly appealing as it offers the lowest cost and the potential use of legacy sources and uncontrolled videos.

However, the video-based facial performance capture is challenging. Despite decades of research in computer graphics and a plethora of approaches, many existing video-based facial capture systems still suffer from three major limitations. First, captured facial models are often extremely coarse and usually only contain sparse collections of 2D or 3D facial landmarks rather than detailed 3D shapes. Second, these results are often vulnerable to ambiguity caused by occlusions, the loss of depth information in the projection from

3D to 2D, and a lack of discernible features on most facial regions and therefore require a significant amount of manual intervention during the capturing process. Third, the previous systems often lack the capability to capture the 3D eye gaze, which is an indispensable component of facial performance.

The goal of this dissertation is to reconstruct 3D facial performances from monocular videos (*e.g.*, Internet videos) automatically and efficiently, enabling capturing fine-scale facial details and advanced applications such as eye gaze tracking and video editing on facial appearance, and overcoming the limitations aforementioned.

This dissertation achieves the goal in three key steps. First, we develop a novel automatic facial feature detection/tracking algorithm that accurately locates important facial features across the entire video sequence. The key idea is a combination the power of local feature detection, Active Appearance Models (AAMs), and temporal coherence. These tracked facial features are then used as the constraint for 3D facial deformation reconstruction, and the accuracy ensures a good alignment between the 3D face model and input video frame. Next, we present a novel automatic high-fidelity facial performance capture method that takes monocular videos as input. It uses the detected facial features along with multilinear facial models to reconstruct 3D head poses and large-scale facial deformation, and uses per-pixel shading cues to add fine-scale surface details such as emerging or disappearing wrinkles and folds. We iterate the reconstruction procedure on large-scale facial geometry and fine-scale facial details to further improve the accuracy of facial reconstruction. Finally, we improve the accuracy and efficiency of the large-scale facial performance capture by introducing a local binary feature based 2D landmark regression and a novel convolutional neural network based pose and expression regression. We complement the facial performance capture system with an efficient 3D eye gaze tracker to achieve real-time 3D eye gaze animation. We have tested our systems on monocular videos downloaded from the Internet, demonstrating the accuracy and robustness under

a variety of uncontrolled lighting conditions and overcoming significant shape differences across individuals.

1.1 Contributions

This dissertation has made the following main contributions to achieve the goal.

- An automatic facial feature detection/tracking framework that accurately locates important facial features across the entire video sequence.
- A novel end-to-end facial performance capture system that automatically reconstructs 3D head poses, large-scale facial deformation, and fine-scale facial details using uncontrolled monocular videos.
- A realtime facial performance capture system that automatically captures 3D head pose, facial deformation, and eye gaze using a monocular video camera.

We also made a set of algorithmic and technical contributions to address all the challenges aforementioned.

- A novel facial reconstruction technique that combines facial detection, non-rigid structure from motion, multilinear facial models, and keyframe based spacetime optimization to compute 3D poses and large-scale facial deformation from monocular video sequences.
- An efficient facial modeling algorithm that infers fine-scale geometric details and unknown incident lighting and face albedo from the whole sequence of input images. Our algorithm builds on the state of the art in 3D face reconstruction from a single image [1]. However, we significantly extend the idea to reconstructing dynamic facial details using monocular videos.

- An efficient convolutional neural network based pose and expression regression framework. It obtains the 3D pose and expression as well as 2D facial features directly from the input face image.

Figure 1.1: Our facial feature detection/tracking algorithm can accurately and quickly track a large number of facial features from RGB/RGBD images, even in the presence of large pose, lighting, skin color, and expression variations as well as occlusions.

1.2 Organization

Figure 1.2: Capturing high-fidelity facial performance automatically using Internet videos and its applications on wrinkle removal and facial geometry editing.

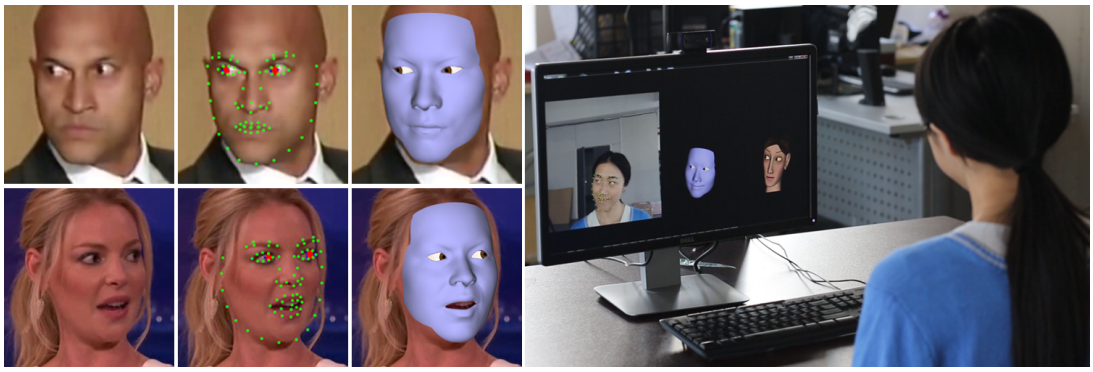


Figure 1.3: Capturing 3D facial and eye gaze performances using monocular video sequences in realtime.

This dissertation is organized as follows. In Section 2, we describe the 2D landmark detection/tracking algorithm that locates important facial features for each video frame (Figure 1.1). It is been applied to RGBD video sequences for realtime tracking. Section 3 presents the high-fidelity facial performance capture system (Figure 1.2) based on the tracked 2D facial features using the algorithm described in Section 2 and the per-pixel shading information. We then introduce our approach for realtime facial performance and 3D eye gaze capture (Figure 1.3) which refines the accuracy and efficiency of the 3D head pose and facial deformation reconstruction and complements it with 3D eye movements in Section 4.

2. ACCURATE ROBUST FACIAL FEATURE DETECTION AND TRACKING USING A SINGLE RGBD CAMERA*

Robust and accurate facial tracking is essential for face modeling, analysis, and recognition. Such a system would also be useful in a variety of applications such as games, human-computer interaction, security, and telepresence. Despite the progress made over the last decade (*e.g.*, [4, 5, 6, 7, 8]), current systems still struggle in tracking person-independent facial expressions in the presence of large pose, lighting, expression, and skin color variations.

In this section, we propose a new method to quickly and accurately detect and track facial features from RGB/RGBD video sequences and apply it to RGBD videos obtained by a single Kinect. The key idea of our approach is to combine the power of local feature detection, Active Appearance Models (AAMs) and temporal coherence for accurate and robust facial detection and tracking. Local detectors and AAMs are complementary to each other. At one end, facial registration using AAMs can achieve sub-pixel accuracy but is often sensitive to the initialization due to the gradient descent optimization. At the other end, local feature detectors can robustly infer the locations of facial features in single images but often with less accurate and unstable results. An appropriate combination of both techniques provides benefits at both ends.

We first introduce an efficient local detector that automatically identifies a small set of facial features in single video images. We formulate local feature detection as a per-pixel classification problem and apply randomized forests to associate each pixel with a probability score of being a particular feature. The outputs of local feature detectors are often

*Part of this section is reprinted with permission from "Accurate and robust 3D facial capture using a single RGBD camera" by YL. Chen, HT. Wu, F. Shi, X. Tong, and J. Chai, in *IEEE International Conference on Computer Vision*, pp. 3615–3622, 2013. Copyright 2013 by IEEE.

noisy and frequently corrupted by outliers due to classification errors. This motivates us to employ geometric hashing to robustly search closest examples in a predefined database of labeled images and use a consensus of non-parametric global shape models to improve the outputs of local detectors. Furthermore, we develop an efficient facial registration method that integrates AAMs, local detection results, and facial priors into a Lucas-Kanade registration framework.

We complement detection with temporal coherence to further improve the robustness and accuracy of our detection/tracking process. In particular, we utilize temporal coherence to predict head poses of the current frame and utilize them to assist our detection process. In addition, we use previously registered facial images to incrementally update the Active Appearance Models (AAMs) on the fly.

We show the system is accurate and robust by comparing against alternative methods. We have also tested our system on a wide range of challenging examples. We show the system can accurately detect and track a large number of facial features even in the presence of large pose, lighting, expression, and skin color variations. Lastly, we validate our method by evaluating the importance of each key component in our system.

2.1 Background

Previous work on facial feature detection and tracking can be approximately classified into two categories: holistic methods and patch-based detection methods.

Holistic methods (*e.g.*, the Active Appearance Models (AAMs) algorithm [4, 5]) construct a combined generative model of shape and texture over the whole face and fit the generative model to a test image by minimizing the texture residual between the test image and the hypothesized image. Such methods, however, suffer from lighting changes, modeling complexity, and a bias towards the average face. They are also sensitive to the initialization and prone to getting stuck in local minima due to gradient-based optimiza-

tion.

An appealing alternative for facial feature detection is to adopt a patch-based representation and assume image observations made for each landmark are conditionally independent. This leads to better generalization with limited data compared to holistic representations, since it needs only to account for local correlations between pixel values. Recently, both generative models [9, 10] and discriminative models [11, 12] have been explored to model image patches centered around the facial features. More recent research has been focused on combining detection results from the various local detectors via global shape constraints over the whole face region, including structured graph models[13, 14], non-parametric global models[15], or a PCA shape model[6, 7, 16].

Our detection method is relevant to Belhumeur et al. [15] because both methods use a consensus of non-parametric global shape models to improve the outputs of the local detectors. However, our approach is different from theirs because we complement local detection with AAMs in Lucas-Kanade registration framework. In addition, our local detector is built on randomized forests rather than support vector machines (SVMs) adopted in their system. Another difference is that we perform KNN search process via geometric hashing rather than RANSAC-based sampling and therefore significantly improve the speed and robustness of the search process. The usage of randomized forests is similar to Dantone et al. [12] and Cootes al. [16]. However, randomized forests are used for regression in their work, while for classification in ours. The simpler binary test enables more flexibility to global rotations, as shown in Section 2.3.1. Besides, our goal is an accurate and robust facial feature tracking system, instead of single frame facial alignment. This leads us to utilize temporal coherence to further improve the accuracy and robustness of our tracking process.

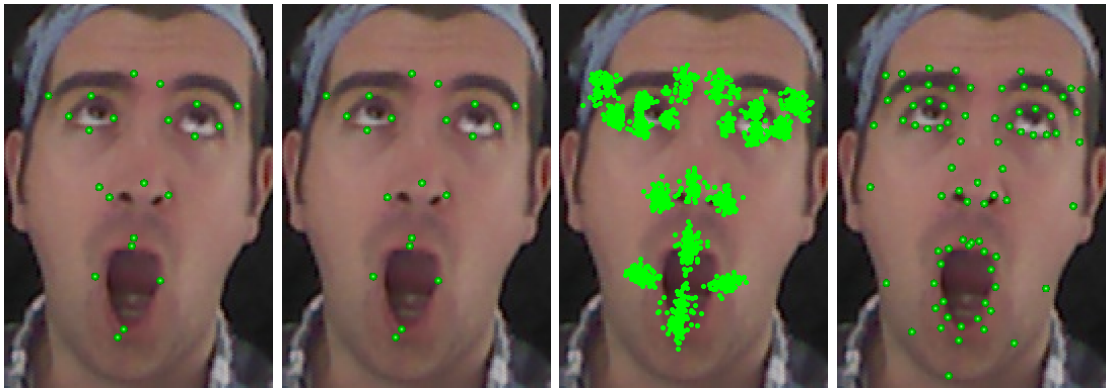
Our work is also related to Baltrusaitis et al. [7], who extended the constrained local model (CLM) [6] to detecting and tracking facial features in RGBD images. Our approach

shares a similar perspective as theirs because both take advantage of color and depth information for facial feature detection and tracking. However, their system is different from ours because they apply SVMs classifiers to the whole RGBD image to get the response maps of facial features and then iteratively update the PCA shape parameters by fitting the local appearances through maximizing a posteriori probability. Our comparison experiment shows our facial feature tracking process outperforms their system running in both single-frame and sequential tracking modes.

Temporal coherence is useful for improving the robustness and accuracy of single-frame facial registration. One common strategy is to initialize facial registration parameters (*e.g.*, AAMs) using registration results obtained from previous frames. Notably, Zhou et al. [17] incorporated local appearance constraints between successive frames as well as results obtained from face segmentation to improve the robustness of AAMs registration process. Our approach, however, is different because it does not explicitly utilize previous registration results for registration initialization. Instead, we utilize a buffer of previously registered images to incrementally update active appearance models (AAMs) on the fly. In addition, we predict in-plane rotations of the head using results from previous frames and utilize prediction to improve the robustness and accuracy of our local detection process.

2.2 Overview

The key idea of our work is to introduce a robust facial detection/tracking algorithm that accurately locates a set of predefined facial features (*e.g.*, the nose tip and the mouth corners) from RGB/RGBD video frames. In Section 2.3 and 2.4, we describe our approach on how to detect facial feature locations in single images. The whole detection process consists of three main steps. The system first applies randomized-forests classifiers and multi-mode detection to detect feature locations in a single image. The features from local detection process are often noisy and frequently corrupted by outliers due to classification



(a) (b) (c) (d)

Figure 2.1: Robust detection of facial features. (a) candidate features after local detection and multi-mode extraction; (b) detected features after outlier removal; (c) closest examples of detected features via geometric hashing; (d) final output.

errors (see Figure 2.1(a)). To handle this challenge, we employ geometric hashing to robustly search closest examples in a training set of labeled images where all the key facial features are labeled, and improve the output of detection process using the closest examples. In the final step, we refine feature locations and extract locations of non-salient facial features by combining detection results with Active Appearance Models (AAMs) and 2D facial priors embedded in closest examples (Figure 2.1(d)). In Section 2.5, we discuss how to complement detection with temporal coherence to further improve the accuracy and robustness of our tracking process. We discuss each step in detail in the following subsections.

2.3 Automatic Feature Detection

In this subsection, we introduce an efficient feature detection process that utilizes local information of a pixel (*i.e.*, an input patch centered at a pixel) to detect a predefined set of facial features from single RGBD images. We take advantage of both color and depth information for facial feature detection. The use of depth information allows our approach to mitigate the effect of lighting conditions and significant scale changes.

2.3.1 Pixel Classifications via Randomized Forests

We formulate the feature detection process as a per-pixel classification problem. During training, we construct a set of $N = 21$ classes of keypoints. Each class corresponds to a prominent facial feature such as the nose tip and the left corner of the mouth. Figure 2.1(b) shows all facial features considered by our local detection process. At runtime, given an input patch centered at an RGBD pixel **x**, we want to decide the likelihood that a particular feature $c \in \{1, ..., N\}$ is located at point **x** in the image.

We use randomized decision trees [18, 19, 20] to train a classifier for automatic labeling of pixels. Randomized trees are an ensemble of L decision trees $T_1, ..., T_L$. Each node in the tree contains a simple test that splits the space of data to be classified (in our case the space of image patches). Each leaf contains an estimate based on training data of the posterior distribution over the classes. A new patch is classified by dropping it down the tree and performing an elementary test at each node that sends it to one side or the other. When it reaches a leaf, it is assigned probabilities of belonging to a class depending on the distribution stored in the leaf. Once the trees $T_1, ..., T_L$ are built, their responses are combined during classification to achieve a better recognition rate than a single tree could.

Specifically, for an input patch centered at pixel **x**, each tree $T_1, ..., T_L$ outputs posterior probabilities $Pr_{\lambda(l,\mathbf{x})}(c|\mathbf{x})$, where c is a label in $C = \{1, ..., N\}$ and $\lambda(l, \mathbf{x})$ is the leaf of tree T_l reached by the patch **x**. The probability of the patch **x** associated with each feature c is the average of the class distributions over the leaf nodes reached for all L trees:

$$Pr(c|\mathbf{x}) = \frac{1}{L} \sum_{l=1,...,L} Pr_{\lambda(l,\mathbf{x})}(c|\mathbf{x}). \tag{2.1}$$

The probability score $Pr(c|\mathbf{x})$ indicates the likelihood that the desired feature c is located at point **x** in the image.

We utilize both color and depth information for pixel classification. Specifically, we constructed two separate randomized forests using color and depth data and use them to evaluate the probability of the patch **x**, resulting in $Pr_{color}(c|\mathbf{x})$ and $Pr_{depth}(c|\mathbf{x})$. The final probability is then the average of the two outputs:

$$Pr(c|\mathbf{x}) = \frac{Pr_{color}(c|\mathbf{x}) + Pr_{depth}(c|\mathbf{x})}{2}. \tag{2.2}$$

For each randomized forest, similar binary tests are performed. The feature function calculates the difference of depth/intensity values of a pair of pixels taken in the neighborhood of the classification pixel. Specifically, at a given pixel **x**, the feature computes

$$f(x) = P(\mathbf{x} + \frac{\mathbf{u}}{D(\mathbf{x})}) - P(\mathbf{x} + \frac{\mathbf{v}}{D(\mathbf{x})}), \tag{2.3}$$

where $P(\mathbf{x})$ is the RGBD value at pixel. The parameters **u** and **v** describe the offsets. Note that the offsets are normalized by corresponding depth values $D(\mathbf{x})$ to achieve depth invariance.

If the value of a splitting function is larger than a threshold, go to the left node and otherwise go to the right node. The choice of which pair of pixels as well as the optimal threshold for splitting the node is automatically determined by maximizing the information gain for particular features during the training phase.

2.3.2 Multiple Mode Extraction

Our next task is to infer feature locations from their probability maps in the whole detection region (or in some smaller region around the face as inferred from an earlier face detection step). One way to achieve this is to extract the location of the highest detector score. In practice, no detector is perfect, so the correct location will not always be at the location with the highest detector score. Instead of choosing feature locations corresponding

to the highest detection score, we extract a set of candidate locations by detecting peaks of all the important modes in probability maps and then utilize prior knowledge embedded in a training set of labeled facial images to remove misclassified features. This idea is motivated by an observation on local feature detection results. We have observed that actual locations of features are always correlated to the peaks of important modes.

We adopt a simple yet very effective method for detecting peaks of important modes. We first process the probability map by excluding all the pixels with scores lower than a threshold ($0.7 * Max(score)$). The remaining pixels are initialized as unmarked. We search the highest detection score among all the unmarked pixels to extract the peak of the first mode. Once the peak of the first mode is extracted, we mark the corresponding pixel as well as its connected regions via flood fill algorithm. We repeat the same process until all the remaining pixels in the detection window are marked. In the last step, we apply meanshift algorithm [21] to refine the location of each extracted mode.

2.3.3 KNN Search by Geometric Hashing

We now discuss how to utilize prior knowledge embedded in a training set of labeled facial images to remove misclassified features. Due to classification errors, feature candidates inevitably contain "outlier" features. Similar to Belhumeur et al. [15], we robustly search closest examples in a training set of labeled images and use them to remove misclassified features. KNN search, however, requires computing the unknown similarity transformations between the detection image and every training image. Instead of adopting an RANSAC-based sampling procedure [15], we propose to use geometric hashing to find the closest examples. Geometric hashing [22] has been successfully applied to 3D object detection and popular for its simplicity and efficiency. The use of geometric hashing for KNN search significantly improves the speed and accuracy of our search process.

In an off-line step, each database example is encoded by treating each pair of fea-

tures as a geometric basis. The remaining features can be represented in an invariant fashion with respect to this basis using two parameters. For each feature, its quantized transformed coordinates are stored in the hash table as a key, and indices of the basis and database examples, denoted as $\{basisID, exampleID\}$, as a value. Then a new pair of basis features is selected, and the process is repeated. In our application, we enumerate all the $N * (N-1)/2$ feature pairs, where $N = 21$ is the number of facial features in labeled facial examples.

In the on-line step, we enumerate all the candidate bases based on the modes extracted from local detection process. For each candidate $basis$, the remaining features are encoded according to the basis and possible correspondences from the database examples are found in the previously constructed hash table. Unlike traditional geometric hashing process, the index of geometric basis (*i.e.* "basisID") in our application is known in advance due to classifications of features. We, therefore, discard all the votes whose $basisID$ is inconsistent with the index of input basis. To better handle outliers from local detection process, we sum up the votes of each database example from all the candidate bases and sort them in a descending order. The top K database examples are then returned as *K* closest examples of the input image. Figure 2.1(c) shows K closest examples of detected features. In our implementation, the number of closest examples is set to $K = 40$. Finally, inliers are selected from all candidate modes using the closest example. If the number of inliers is smaller than a specific threshold (6 in our experiment), the whole process stops and returns as failed. The key steps of our KNN search process are described in Algorithm 1.

2.4 Feature Detection Refinement

This step is necessary for accurate feature detection because of two reasons. First, even with outlier removal, results obtained from feature detection are often noisy. The evalu-

Algorithm 1 KNN search by Geometric Hashing

Input: Candidate modes for N features, Hash-table h which is built on an RGB database with the size of M frames

Output: The indices for K closest examples

Initialize the votes for all the training examples $v(m), m = 1, ..., M$ to 0
for each basis b_i, $i = 1, ..., N * (N-1)/2$ **do**
  Calculate quantized transformed coordinates $c = \{c_1, ...c_{N-2}\}$ for the remaining $N-2$ modes
  for each c_j, $j = 1, ..., N-2$ **do**
    Obtain a set of votes $\{basisID, exampleID\}$ from previously constructed hash table $h(c_j)$
    for each vote if $basisID == i$ **do**
      The corresponding $v(exampleID) + +$;
    end for
  end for
end for
Sort $v(m), m = 1, ..., M$ in a descending order
return the first K indices of sorted $v(m), m = 1, ..., M$

ation on feature detection process confirms this concern (see Figure 2.8). Second, similar to previous local detection algorithms, our facial feature detection process is mainly focused on detecting salient features and therefore might not be sufficient for some applications such as animation, gaming, and human computer interaction. To address this challenge, we propose to refine feature detection results by complementing detection with facial alignment using Active Appearance Models (AAM) [23]. Figure 2.1(b) and (d) show the improvement of feature locations as well as detection of non-salient facial features via the refinement step.

We formulate the refinement process in an optimization framework. The whole cost function consists of three terms:

$$E = w_1 E_{AAM} + w_2 E_{detection} + w_3 E_{prior}, \tag{2.4}$$

where the first term E_{AAM} is the AAM term, which measures the inconsistency between the input image and the AAM model instance. The second term is the *detection* term, which penalizes the deviation of feature points from detected feature points from Section 2.3.3. The third term is the *prior* term, which ensures the new feature points are consistent with 2D facial priors embedded in *K* closest examples. In this work, we fit a Gaussian prior based on *K* closest examples and obtain this term by applying the negative log to the Gaussian distribution. The local priors reduce bias towards the average face and avoid the problem of finding an appropriate structure for global priors, which would necessarily be high-dimensional and nonlinear.

2.4.1 Optimization

We minimize the cost function by simultaneously optimizing the shape and appearance parameters of the AAM model instance, as well as the global similarity transformation for aligning the input image with the AAM model instance. This requires minimizing a sum of squared nonlinear function values. We analytically derive Jacobian and Hessian of each objective term and optimize the function in Lucas-Kanade registration framework via iterative linear system solvers [23]. Lucas-Kanade algorithm, which is a Gauss-Newton gradient descent nonlinear optimization algorithm, assumes that a current estimate of parameters is known and then iteratively solves for increments to the parameters using linear system solvers.

We initialize the shape parameter using the closest example of the input image. The initialization of the similarity transformation is obtained from detection process via KNN search. The appearance parameters are initialized by the average appearance image of AAM models. The optimization typically converges in 8 iterations because of very good initialization and local facial priors obtained from *K* closest examples. We set the weights of w_1, w_2 and w_3 to 2, 1, and 0.001 respectively. During the iterations, we gradually

decrease the weight for the first term (w_1) from 2 to 0.0001 in order to ensure that the final feature locations can achieve a better accuracy via AAM fitting.

2.4.2 Handling Significant Occlusions

When significant occlusions occur, incorporating the AAM term into facial detection might deteriorate the performance of detection process. This is because occlusions result in a large fitting residual between the input image and model instance. The system automatically monitors the status of facial alignment. Once it detects the "failure" mode (*i.e.*, when the residual of AAM fitting term is larger than a user-specified threshold (0.6)), the system switches back to detection mode and returns the closest example as the detection output.

2.4.3 Realtime GPU Implementation

The fact that each step in our facial alignment algorithm can be executed in parallel allows implementing a fast solver on modern graphics hardware. By using CUDA to implement our tracking algorithm, the current system runs in real time (38 frames per second) on a machine with Intel Core i7 3.40GHz CPU and GeForce GTX 580 graphics card. The typical computational times for detection using randomized forests, KNN via geometric hashing, and optimization are 10ms, 1.2ms, and 15ms respectively.

2.5 Incorporating Temporal Coherence

Single frame feature detection can automatically infer the locations of facial features from single RGBD images but often with noisy and unstable tracking results. In addition, our facial detector often fails to locate facial features when the subjects are under large roll angles because the current training database does not include training examples corresponding to large roll angles. Another issue is that our detection refinement process builds upon AAMs and therefore might not generalize well to new subjects significantly different

from training databases. In the following, we discuss how to complement facial detection with temporal coherence to improve the robustness and accuracy of our facial detection and tracking process.

2.5.1 Adaptive AAMs

Our first idea is to utilize previously registered facial images to incrementally update the Active Appearance Models (AAMs) on the fly. During tracking, we maintain a buffer of registered facial images from previous frames and use them to incrementally update the mean and eigen basis of AAMs based on an incremental learning method proposed by Ross et al. [24]. Note that we only push the registered frames whose corresponding AAM fitting errors are below a particular threshold (0.6) into the buffer. Once the buffer is full, we first check if the registered images in the buffer are sufficiently far from the subspace of the current AAMs. When the reconstruction residual is higher than a threshold (0.3), we update the AAMs and then reinitialize the whole buffer. In our experiment, we set the buffer size to 10.

2.5.2 Dealing with Large In-plane Rotations

The temporal coherence can also be utilized for improving the tracking robustness and accuracy with respect to large head poses. In particular, we use the roll angles from previous frames to predict the current roll angle of the head and then apply the predicted roll angle to transform/normalize the current RGBD images. The normalized RGBD images can then be fed into randomized trees for facial feature detection. The normalization ensures the classification features are invariant to in-plane rotations. Again, we utilize the temporal coherence only when the previous frames are well registered (*i.e.*, when AAM fitting error is below 0.6).

2.6 Evaluation and Experiment

We have conducted three types of experiments to evaluate the effectiveness of our algorithm. We first show our algorithm is accurate and robust by comparing against alternative methods for facial feature tracking using a single RGBD camera. We then validate our method by evaluating the importance of each key component in our system. Finally, we test our method on *Biwi Kinect* head pose dataset [25] and demonstrate the robustness and accuracy of our facial detection and tracking process in the presence of large pose variations. The random forests are trained on 97 RGBD sequences taken by Kinect, which contain variations in identities, expressions, head poses, and illuminations. 1637 RGB images sampled from the video sequences and the LFPW database [15] are used for AAM training.

2.6.1 Comparisons Against Alternative Methods

We compare our facial tracking algorithm against CLM [6], CLM-Z [7] and Microsoft Kinect facial SDK [8] running in both single frame detection mode and sequential tracking mode. We evaluate the algorithms based on seven RGBD image sequences taken from seven different subjects (Figure 2.6). Note that all subjects for testing are excluded from the training databases. The seven testing sequences consist of 3341 frames in total and include variations caused by occlusions and differences in facial expressions, head poses, illuminations, and skin colors. We used a semi-automatic technique to annotate the ground truth feature locations for each sequence. Specifically, for each testing sequence, we first identify a small number of key frames, manually label facial feature locations in each frame, and use them to build AAM to track facial features across the entire sequence. We continue adding key frames, refining locations of the facial features, and doing AAM tracking until a satisfactory annotation result is achieved.

We evaluate the error of a feature detection/tracking algorithm by computing the aver-

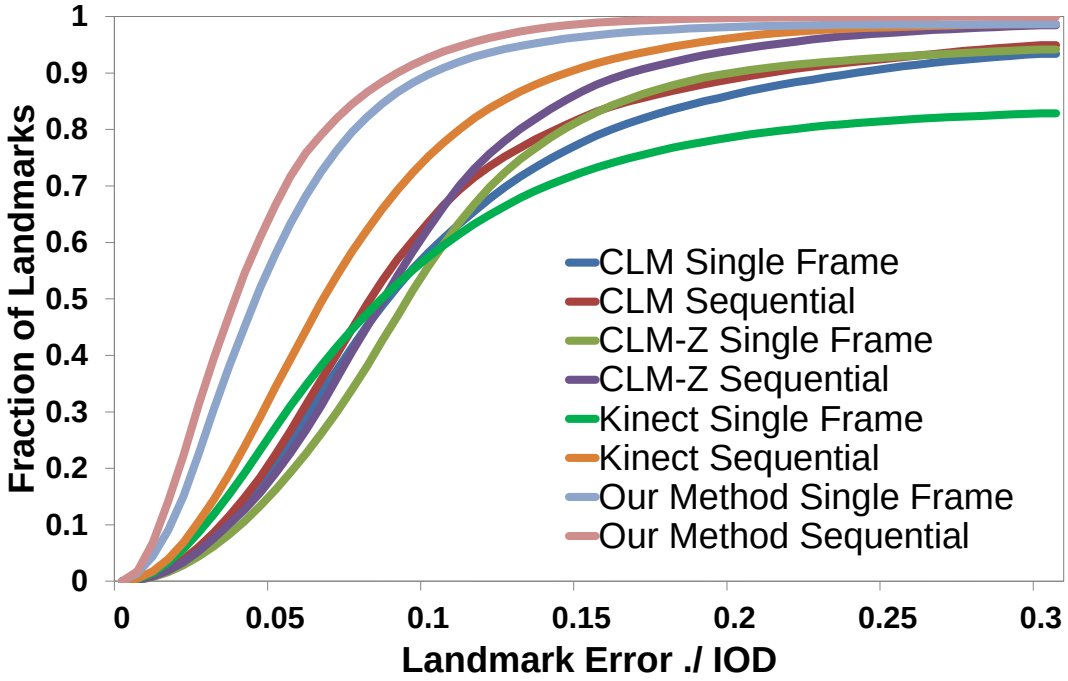


Figure 2.2: Comparison against alternative methods: cumulative error curves based on seven testing RGBD image sequences.

age distance between detected feature locations and the ground truth locations. To make it invariant to face size, we normalize the error in pixels by the inter-ocular distance (*i.e.*, the distance between the two pupils), similar to most previous work in this area. We exclude occluded facial features from evaluation because it is hard to accurately annotate the ground truth location of invisible features. For some methods such as CLM-Z [7] and Kinect facial SDK [8], the system occasionally fails to output any results. When this occurs, we set the error value of each facial feature to 0.35.

There are slight differences of facial feature sets adopted in different systems. To make a consistent comparison for all the methods, we define a common set of facial landmarks and obtain their locations based on detected facial features (Figure 2.4). Note that for Kinect facial SDK and our method, we calculate the locations of facial features on the nose bridge by averaging the locations of their corresponding features on the left and right

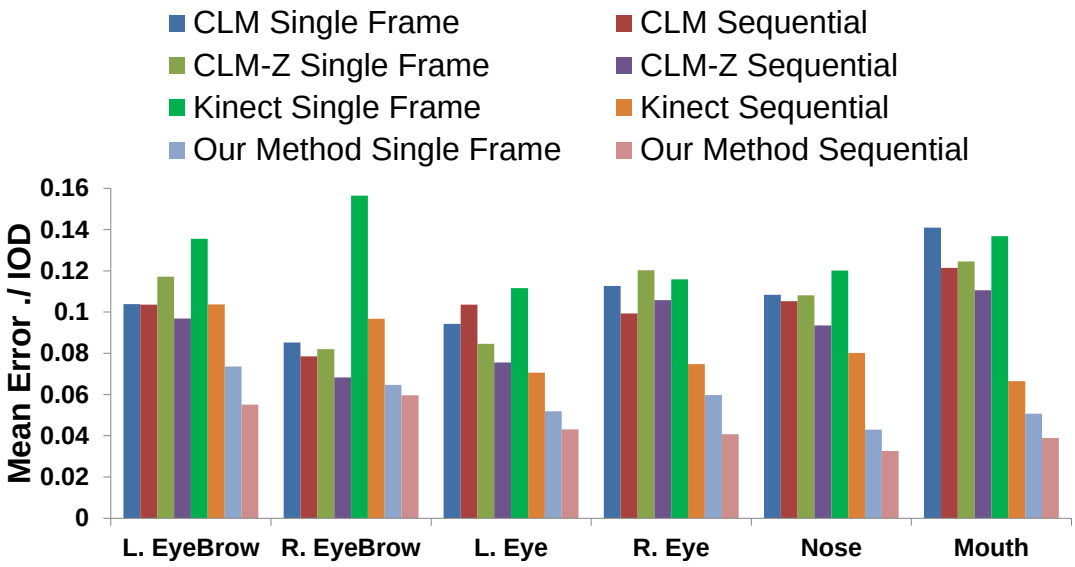
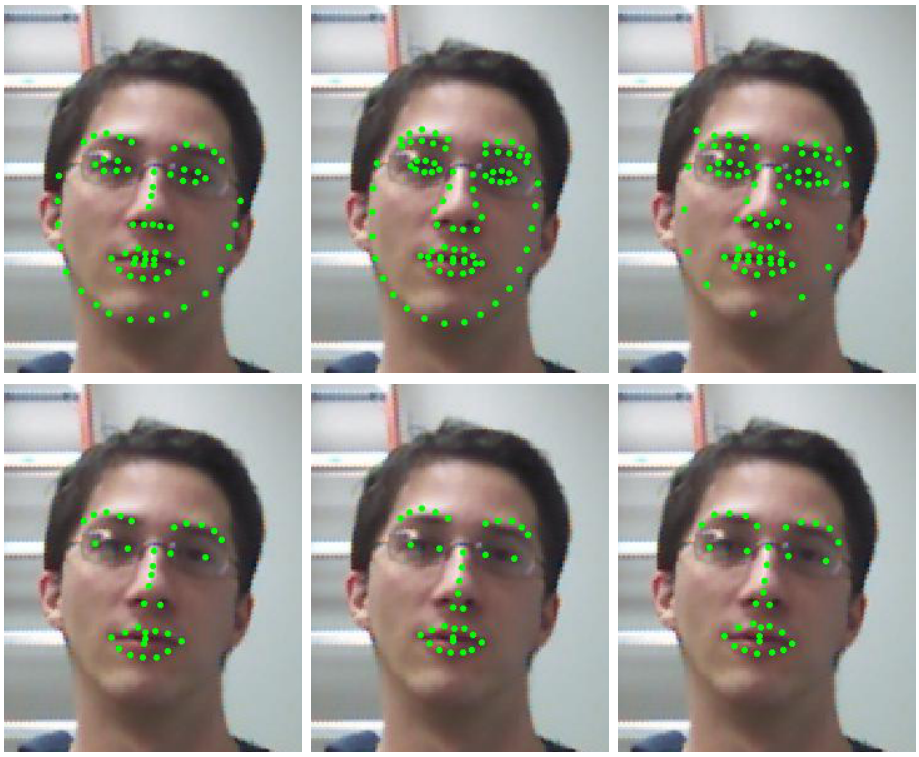


Figure 2.3: Comparison against alternative methods: mean error for each facial region.

side of the nose bridge. Also, similar to work [15, 26], fixed offsets for each landmark are applied to further reduce the differences of feature definitions among the different methods.

Figure 2.2 compares our facial detection/tracking method against alternative methods [6, 7, 8] in both single frame detection mode and sequential tracking mode. We also report the tracking errors corresponding to six facial regions (see Figure 2.3). The comparison clearly shows that our method with/without temporal coherence significantly outperforms other methods running in either single frame detection mode or sequential tracking mode. Figure 2.6 shows sample frames of comparison results.

The importance of temporal coherence. As described in Section 2.5, we utilize temporal coherence to further improve the accuracy and robustness of our detection/tracking process. The last two columns of Figure 2.6 show some sample frames for comparisons between our single-frame detection and sequential tracking process. The results show that large head rotations and significant appearance changes can be better handled in the



CLM(-Z) Kinect Ours

Figure 2.4: Top row: different feature settings of the four methods. Bottom row: the common feature set used for evaluation. Note that for Kinect facial SDK and our method, we calculate the locations of three facial features on the nose bridge by averaging the locations of their corresponding features on the left and right side of the nose bridge.

sequential tracking mode. In addition, Figure 2.2 and Figure 2.3 show that that better accuracy is achieved by using temporal coherence.

Figure 2.5 confirms the benefit of adaptive AAMs modeling. It compares the registration errors of AAM term for a test sequence with 625 frames with/without adaptive AAMs. As shown in the figure, facial tracking using adaptive AAMs produces much smaller registration errors for the AAMs term. The registration error drops rapidly after the first AAM model update and remains smaller across the entire sequence.

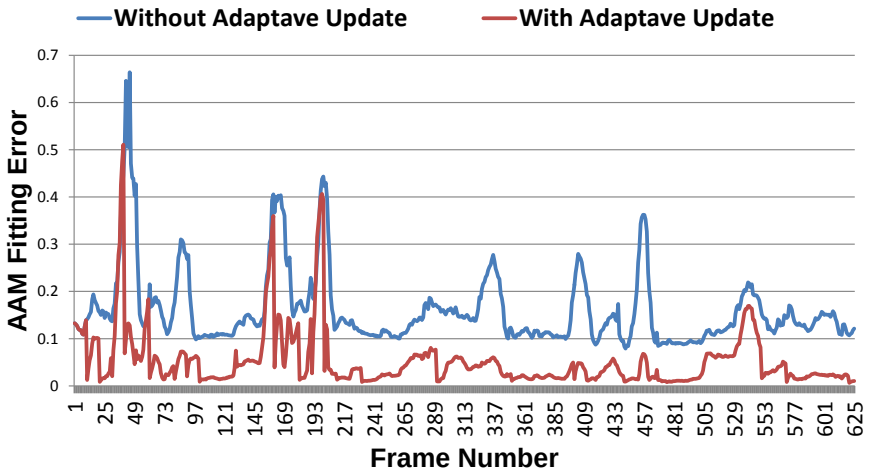
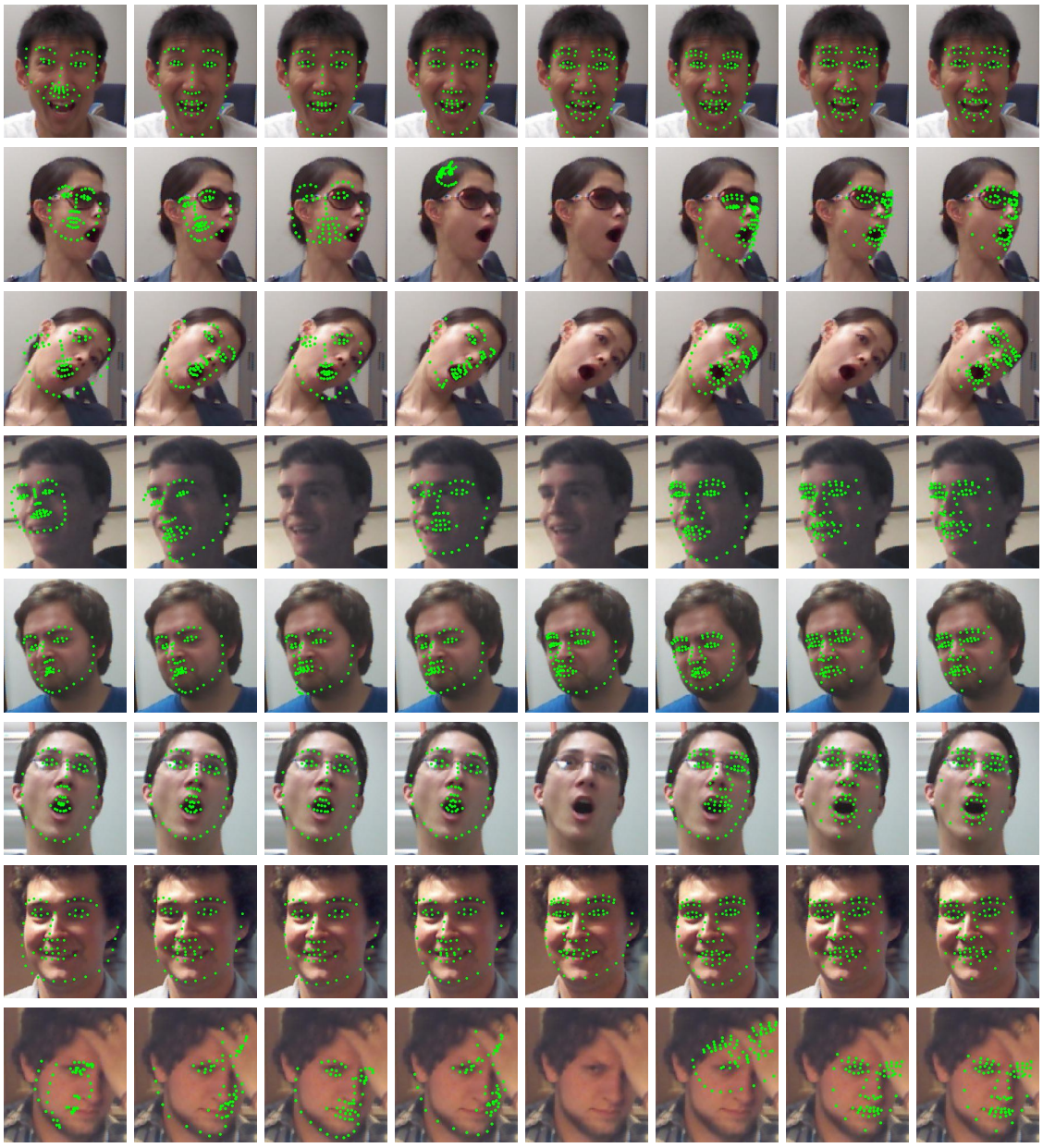


Figure 2.5: Active Appearance Models (AAM) fitting errors with/without adaptive updates on AAM.

2.6.2 Evaluation of Our Detection Method

We now evaluate the key component of our single frame detection process using single RGBD test images. We first demonstrate the benefit of using both color and depth information for facial feature registration. We further show the necessity of combing local detection with AAMs by dropping off each term in the cost function described in Equation 2.4. The evaluation is based on cross validation on RGBD/RGB training data sets. Specifically, for each test subject, we pulled all the training data of the test subject out of the training databases and used the training data sets of other subjects to learn randomized-forests classifiers and AAM. The excluded RGBD images are then used for testing. The final error is computed based on 317 testing RGBD images from 19 subjects.

Evaluation of detection term. Our local detection algorithm utilizes both color and depth information for facial feature detection. This evaluation compares facial detection results based on color information, depth information, and a combination of color and depth information. Figure 2.7 shows the combination of both color and depth information achieves a better detection accuracy.



(a) (b) (c) (d) (e) (f) (g) (h)

Figure 2.6: Comparison against alternative methods. (a) Constrained Local Model (CLM) (per-frame); (b) CLM (sequential); (c) CLM-Z (per-frame); (d) CLM-Z (sequential); (e) Kinect facial SDK (per-frame); (f) Kinect SDK (sequential); (g) Our method (per-frame); (h) Our method (sequential). Note that no markers are drawn if the method does not return a result.

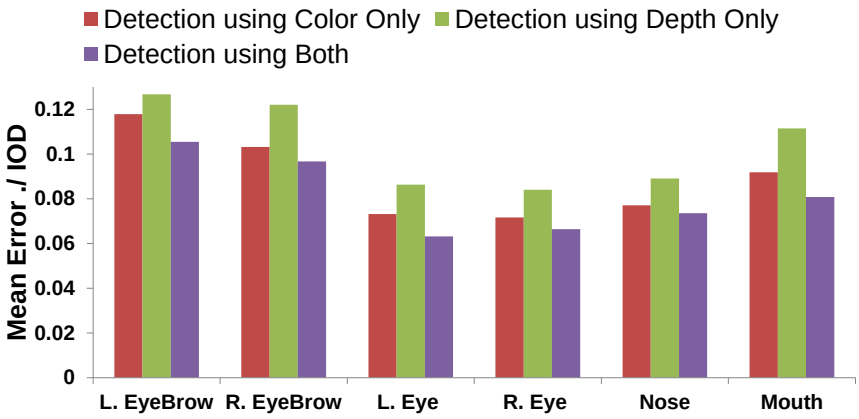


Figure 2.7: Comparisons of mean errors from detection using color information, detection using depth information, and detection using both information.

The importance of each term. We now evaluate the importance of each term in our objective defined in Equation 2.4. We compare results from "detection using color images", "detection using depth images", "detection using both color and depth images", AAM, and our method. All the evaluations are based on single frame detection mode. For AAM term evaluation, we initialize AAM using mean shape and texture. To make a fair comparison, we also initialize the global transformation of AAM using ground truth data.

Figure 2.8 shows the cumulative error curves for each method. Per-frame AAM algorithm, even with ground truth global transformation, often fails to produce accurate results when the actual feature locations are far away from their initial positions. In contrast, detection methods can robustly detect feature locations in single RGBD images but the results lack detailed accuracy. Our detection method combines the power of AAM and detection, and can robustly identify feature locations with the highest accuracy.

2.6.3 Test on Biwi Dataset

For completeness, we also evaluate the robustness and accuracy of our tracking method with large head pose changes using Biwi Kinect pose dataset [25], which contains 20 pose

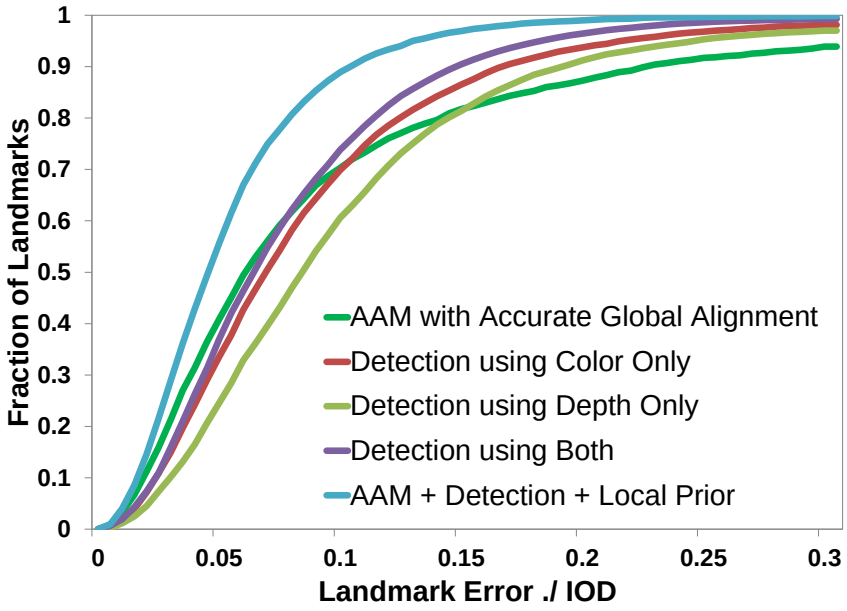


Figure 2.8: Evaluation of the importance of each component in our method. Cumulative error curves show the necessity of combing detection and AAM.

changing sequences with few expressions (almost neutral). We use the same model in the evaluation and run our facial tracking algorithm on the same 4 sequences as in CLM-Z [7]. We compare the accuracy of the shared features with the same ground truth used in their work. Table 2.1 shows the average converged rate and mean error for both methods. Note that the comparison is done without tuning any offset. Figure 2.9 shows our method can track facial features with large pose variations. CLM-Z [7] has a higher convergence rate for large pose variations as it aims to track both rigid facial tracking and nonrigid facial tracking while our system is focused on nonrigid facial tracking only. On the other hand, unlike CLM-Z [7], our system does not require a person specific 3D mesh model for rigid facial tracking and therefore is more flexible. It is also worth pointing out that the current training data sets for our system have limited pose variations in yaw and pitch and almost no pose variations in roll. We believe the convergence rate of the system can be further improved by including training data sets with more head pose variations, which will be

Figure 2.9: Testing results on Biwi dataset. Our experiment shows that the system can track facial features with large pose variations. According to the ground truth annotation, our method covers the a large pose range (54° in yaw and 26° in pitch).

part of our future work.

Method	Converged	Mean Error /IOD
CLM-Z	79%	0.125
Ours	70%	0.084

Table 2.1: Comparison with CLM-Z on Biwi dataset. The mean errors are calculated only for converged frames (< 0.3 of the pupil distance).

2.7 Discussion

In this section, we have presented a new algorithm that quickly and accurately detecting and tracking facial features from video frames, and applied it to RGBD images obtained by a Kinect camera. Our approach combines the advantages of local detection, AAMs, and temporal coherence and achieves accurate and robust results. The usage of local detection provides good initializations as well as a proper local shape prior, and the AAMs fitting term ensures the accuracy by synthesizing a face image that is as close to the input face as possible. We integrate the local detection term, local prior term, and the AAMs term into a Lucas-Kanade registration framework. The temporal coherence is also added to predict the head pose and refine the AAMs on the fly. The whole framework is

robust, fully automatic, and can accurately track a large number of facial features even in the presence of large pose, lighting, and expression variations as well as occlusions.

Based the accurately detected/tracked 2D facial features, we are now able to build the 3D facial performance capture system. The key idea is to estimate the pose, camera parameters as well as large-scale facial deformation so that the projected facial features best fit the tracked ones. It is also ideal to recover fine-scale facial details which are often missing in the current facial performance capture systems. This leads us to our next work on high-fidelity facial performance capture and animation from monocular videos (Section 3).

3. AUTOMATIC HIGH-FIDELITY FACIAL PERFORMANCE CAPTURE USING MONOCULAR VIDEOS*

In this section, we present an automatic technique for acquiring high-fidelity facial performances using monocular video sequences such as Internet videos. The key idea of our approach is to use both high-level facial features and per-pixel shading cues to reconstruct 3D head poses, large-scale deformations, and fine-scale facial details in a spacetime optimization framework. We start the process by automatically detecting/tracking important facial features such as the nose tip and mouth corners across the entire sequence. The detected facial features are then used to reconstruct 3D head poses and large-scale deformations of a detailed face model at each frame. This step combines the power of non-rigid structure from motion, multilinear facial models, and keyframe based spacetime optimization for large-scale deformation reconstruction. Next, we utilize per-pixel shading cues to add fine-scale surface details such as emerging or disappearing wrinkles and folds into large-scale facial deformations. Finally, we iterate our reconstruction procedure on large-scale facial geometry and fine-scale facial details to further improve the accuracy of facial reconstruction.

Our final system is robust and fully automatic, allowing for high-fidelity facial performance capture of large-scale deformation and fine-scale facial detail. We have tested our system on monocular videos downloaded from the Internet, demonstrating its accuracy and robustness under a variety of uncontrolled lighting conditions and overcoming significant shape differences across individuals.

*Reprinted with permission from “Automatic acquisition of high-fidelity hacial performances using monocular videos" by F. Shi, HT. Wu, X. Tong, and J. Chai, *ACM Transactions on Graphics*, vol. 33, no. 6, p. 222, 2014. DOI: 10.1145/2661229.2661290. Copyright 2014 by ACM, Inc.

3.1 Background

Our system automatically captures high-fidelity facial performances using monocular video sequences. Therefore, we focus our discussion on methods and systems developed for acquiring 3D facial performances.

One of the most successful approaches for facial capture is based on marker-based motion capture systems [27], which robustly and accurately track a sparse set of markers attached to the face. Recent efforts in this area (*e.g.* [28, 29]) have been focused on complementing marker-based systems with other types of capturing devices such as video cameras and/or 3D scanners to improve the resolution and details of reconstructed facial geometry. Marker-based motion capture, however, is expensive and cumbersome for 3D facial performance capture.

Marker-less facial performance capture provides an appealing alternative because it is non-intrusive and does not impede the subject's ability to perform facial expressions. One solution to the problem of marker-less facial capture is the use of depth and/or color data obtained from structured light systems [30, 31, 32, 33]. For example, Zhang and colleagues [30] captured 3D facial geometry and texture over time and built the correspondences across all the facial geometries by deforming a generic face template to fit the acquired depth data using optical flow computed from image sequences. Ma et al. [31] achieved high-resolution facial reconstructions by interleaving structured light with spherical gradient photometric stereo using the USC Light Stage. Recently, Li and his colleagues [32] captured dynamic depth maps with their realtime structured light system and fitted a smooth template to the captured depth maps.

Reconstructing high-quality face models directly from multiview images offers another possibility for marker-less motion capture [34, 35, 36, 37]. In particular, Bradley and his colleagues [34] used multi-view stereo reconstruction techniques to obtain initial

facial geometry, which was then used to capture 3D facial movement by tracking the geometry and texture over time. Beeler et al. [35] presented an impressive multi-view stereo reconstruction system for capturing the 3D geometry of a face in a single shot and later extended the method to acquiring dynamic facial expressions using multiple synchronized cameras [36]. More recently, Valgaerts et al. [37] combined image-based tracking with shading-based geometry refinement to reconstruct facial performances from stereo image sequences.

The minimal requirement of a single camera for facial performance capture is particularly appealing as it offers the lowest cost and a simplified setup. However, the use of a single RGB camera for facial capture is often vulnerable to ambiguity caused by the loss of depth information in the projection from 3D to 2D and a lack of discernible features on most facial regions. One way to address the issue is to use person-specific facial prior models to reduce reconstruction ambiguity (*e.g.*, [38, 39]). However, fine face details such as wrinkles and large lines cannot be recovered with this approach. In addition, their tracking process is often performed in a sequential manner and therefore requires good initialization and manual correction for troublesome frames.

Recently, Cao and colleagues [40] proposed a 3D regression algorithm that utilized personalized blendshape models for automatic, realtime facial tracking/retargeting. Their approach, however, required an expensive offline training stage to construct person-specific blendshape models. In addition, they focused on tracking large-scale geometric deformation rather than authentic reconstruction of high-fidelity facial performances. Suwajanakorn and colleagues [41] proposed a dense 3D flow algorithm coupled with shape-from-shading to reconstruct high-fidelity facial geometry from monocular videos. Though their method did not require person-specific scan/blendshapes, it used a photo gallery of the subject's faces under different illuminations for reconstructing a person-specific average facial model, which could be unavailable for subjects in uncontrolled videos. Addi-

tionally, instead of using a person-specific average facial model for facial capture, we propose to use multilinear face models to reduce the reconstruction ambiguity of facial capture, thereby significantly improving the robustness of the system. Finally, their method assumed a known albedo map obtained from the subject's photo gallery and estimated lighting for every frame separately. In contrast, we use the whole sequence of input images to estimate a single lighting map and face albedo and therefore further improve the accuracy of facial reconstruction.

Among all the systems, our work is most closely related to Garrido et al. [42], which captured detailed, dynamic 3D facial geometry using monocular video sequences. Briefly, they first created a personalized blendshape model for the captured actor by transferring the blendshapes of a generic model to a single static 3D face scan of the subject. They then tracked 2D image features across the entire sequence by combining sparse facial feature tracking and optical flow estimation. At a final step, they reconstructed fine-scale facial detail by estimating the unknown lighting and exploiting shading for shape refinement.

Our research shares a similar goal but there are important differences. Garrido et al. [42] relied on a manual specification of correspondences between a generic blendshape model and a static face scan of the subject to reconstruct a personalized blendshape model for the subject. In addition, their optical-flow based tracking process required manual intervention to improve the locations of 2D features in the first frame for texturing. In contrast, we automatically detect facial features across the entire sequence and use them along with multilinear facial models to simultaneously compute 3D head poses and large-scale deformations in a keyframe based optimization framework. Garrido et al. reported that typical manual intervention for a test video sequence required about 40 minutes, while our method is fully automatic. In addition, they required a static 3D face scan of the subject for transferring a generic blendshape model to the subject. Our system does not have such a limitation and therefore can be applied to capturing high-fidelity facial performances

from uncontrolled monocular videos such as Internet videos. Finally, we have compared against their method and the comparison shows that our system produces more accurate results than theirs (Section 3.7.2).

Our work on fine-scale detail reconstruction builds on the success of modeling fine-scale facial geometry from a single image proposed by Kemelmacher-Shlizerman and Basri [1]. In particular, they introduced a novel method for shape recovery of a face from a single image by using a reference 3D face model and a reference albedo. We present three novel extensions to their work. First, we extend their idea to shape recovery of a dynamic face from a monocular video sequence. We reduce the ambiguity of estimating lighting and albedo by utilizing the assumption that the lighting and albedo are consistent across the entire sequence. This assumption allows us to estimate the unknown lighting coefficients and albedo based on shading cues across the entire sequence. Second, we utilize results obtained from large-scale deformation reconstruction to initialize and guide the fine-scale geometry reconstruction process, thereby significantly improving the accuracy, robustness, and speed of the process. Third, they simplified the reconstruction problem with linear approximations and directly reconstructed depth in a least-squares fitting framework. In contrast, we propose a two-step optimization algorithm to sequentially compute normal map and depth. As shown in Section 3.7.2, our system produces more accurate results than their method.

The idea of using spherical harmonics approximation for fine-scale detail recovery is similar to previous methods proposed by Wu et al. [43] and Valgaerts et al. [37]. Our method, however, is different from theirs. First, we assume consistent lighting and albedo across the entire sequence while they estimate both maps for each frame. Additionally, we estimate a per-pixel albedo map while they assume it is piecewise uniform. Finally, we measure the differences between the synthesized and observed images based on RGB intensities rather than high-frequency components (image gradients) because we also want

to use recovered surface details to refine large-scale facial deformation.

Our system is also relevant to previous work on tracking 3D facial expression using a single RGBD camera such as Microsoft Kinect or time-of-flight (TOF) cameras [44, 45, 46, 47]. Notably, Weise et al. [44] used RGBD image data captured by a single Kinect and a facial 3D template, along with a set of predefined blendshape models, to track facial expression over time. Most recently, Bouaziz et al [45] and Li et al. [46] concurrently developed realtime monocular face trackers based on a run time shape correction strategy for combined depth and video data. All these systems are focused on modeling large-scale facial deformation rather than high-fidelity facial performances. In addition, they are based on depth and image data obtained by a calibrated RGBD camera rather than RGB images captured by an uncalibrated video camera. It is not clear how their approaches can be extended to capture high-fidelity facial performances from uncontrolled monocular videos.

3.2 Overview

Our system acquires high-fidelity facial performances from uncontrolled monocular video sequences. The problem is challenging because of complex facial movements at different scales, ambiguity caused by the loss of depth information in the projection from 3D to 2D, and a lack of discernible features on most facial regions. Unknown camera parameters and lighting conditions further complicate the reconstruction problem. To this end, we decompose high-fidelity facial performances into three scales: high-level facial features, large-scale facial deformations, and fine-scale facial details, and reconstruct them from coarse to fine scales. We start with the coarse scale (high-level facial features). During the reconstruction, we utilize the results in coarse scales to initialize and guide the reconstruction in fine scales. At a final step, we iterate our reconstruction procedure on large-scale facial geometry and fine-scale facial detail to obtain the final output. The

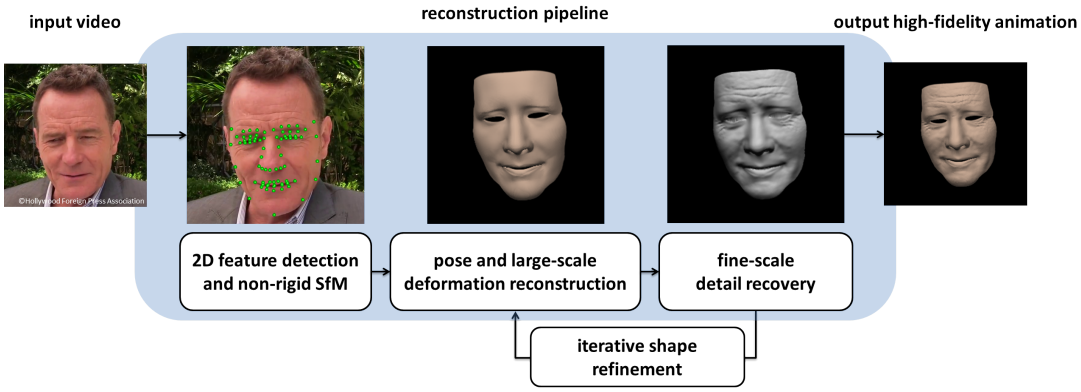


Figure 3.1: Our high-fidelity facial performance reconstruction pipeline. We first detect and track a sparse set of 2D facial features and recover their 3D positions and unknown camera parameters using non-rigid SfM techniques. We then reconstruct 3D head poses and large-scale facial geometry using the estimated facial features and camera parameters. We assume that the lighting and albedo are consistent across the entire sequence and extend shape-from-shading techniques to reconstruct fine-scale details throughout the whole sequence. Lastly, we iteratively refine large-scale facial deformations and fine-scale facial details to obtain the final result.

whole system consists of four main components summarized as follows (Figure 3.1).

The first component is facial feature detection and 3D reconstruction. We start the process by automatically detecting and tracking important facial features such as nose tip, eye and mouth corners in monocular video sequences. We apply non-rigid factorization techniques to recover 3D feature positions and unknown camera parameters, which we use to initialize and guide the large-scale deformation reconstruction process.

The second component is large-scale deformation reconstruction. Sparse facial features, however, do not provide detailed facial geometry. We introduce an efficient reconstruction process that utilizes the recovered 3D facial features and camera parameters, along with multilinear facial models, to model detailed facial geometry. We formulate it as a spacetime optimization problem by simultaneously reconstructing head poses and large-scale facial geometry across the entire sequence. This, however, requires solving

a challenging nonlinear optimization with a huge number of unknowns. We address the challenge by developing a keyframe based optimization algorithm.

The third component is fine-scale detail recovery. Recovering dynamic geometric details such as wrinkles and folds is crucial for high-fidelity facial performance capture. We have developed an efficient shape-from-shading algorithm that infers fine-scale geometric details, the unknown incident lighting, and face albedo from the whole sequence of input images. Our method assumes both lighting and face albedo are consistent throughout the whole sequence. Starting from large-scale deformation results, we simultaneously compute per-pixel normal map of each input image, unknown incident lighting, and face albedo by minimizing the inconsistency between the rendered and observed image sequences. We further reconstruct a per-pixel depth estimate from the reconstructed per-pixel normal map. Again, we use keyframe based optimization to facilitate the reconstruction of fine-scale facial details.

The fourth component is iterative shape refinement. Large-scale facial geometry reconstructed from sparse facial features often does not closely fit actual facial geometry because of the lack of facial features in some facial regions such as cheeks. We address the issue by iteratively refining large-scale facial geometry using the reconstructed per-pixel normal map and then updating the per-pixel normal map with the refined large-scale deformation.

We describe these components in detail in the following subsections.

3.3 Feature Detection and Non-rigid SfM

Our first challenge is how to reconstruct 3D facial feature locations from uncalibrated monocular video sequences. This is achieved by detecting/tracking a sparse set of facial features across the entire sequence using the framework proposed in the previous section and performing non-rigid SfM on the tracked 2D features. The non-rigid structure

from motion step aims to reconstruct 3D feature positions and unknown camera parameters across the entire sequence, which requires solving a non-rigid structure from motion problem. Our solution is based on a prior-free non-rigid structure from motion method proposed by Dai et al. [48]. We choose their method because it is purely convex, very easy to implement, and is guaranteed to converge to an optimal solution. By assuming a weak perspective camera model, the non-rigid structure from motion process reconstructs camera motion and non-rigid shapes through an SVD based factorization.

3.4 Reconstructing 3D Pose and Large-scale Deformation

This subsection describes our idea on how to reconstruct large-scale facial geometry and 3D head poses from 2D feature locations obtained from Section 3.3. We formulate this in a spacetime optimization framework and simultaneously reconstruct large-scale deformations and 3D head poses across the entire sequence. A direct estimate of large-scale deformations and 3D head poses throughout the whole sequence is often time-consuming and memory intensive. This problem motivates us to develop a keyframe based optimization method to speed up the optimization process.

3.4.1 Representation and Formulation

We model large-scale facial deformation using multilinear facial models [39, 40]. The large-scale facial deformation is parameterized using two low-dimensional vectors controlling "identity" and "expression" variation. As a result, we can represent large-scale facial geometry of the subject at any frame using

$$M = R(C_r \times_2 m_{id}^T \times_3 m_{exp}^T) + T, \tag{3.1}$$

where M represents large-scale facial geometry of an unknown subject. And R and T represent the global rotation and translation of the subject. C_r is the dimension-reduced

core tensor, and m_{id} and m_{exp} are identity and expression parameters respectively. Our multilinear model is constructed from FaceWarehouse [49], which contains face meshes corresponding to 150 identities and 47 facial expressions. In our experiment, the numbers of dimensions for the identity and expression parameters are set to 50 and 25.

We assume that the camera projection is weak perspective. The relationship between a 2D facial feature $\mathbf{p}_k$ and its corresponding large scale facial geometry model can be described as follows:

$$\mathbf{p}_k = sR((C_r \times_2 m_{id}^T \times_3 m_{\text{exp}}^T)^{(k)}) + \mathbf{t}, \tag{3.2}$$

where s is the scalar for the weak perspective projection and $\mathbf{t}$ represents the translation components on the image space. Note that t_z in T is dropped because of the weak perspective camera model assumption.

Our goal is to estimate a number of unknown parameters ($\{R, \mathbf{t}, s, m_{id}, m_{\text{exp}}\}_j$) across the entire sequence $j = 1, ..., N$. Since the identity is the same throughout the whole sequence, the parameters to be estimated in large-scale deformation reconstruction are $m_{id}, \{R, \mathbf{t}, s, m_{\text{exp}}\}_j, j = 1, ..., N$.

We formulate large-scale deformation reconstruction in a spacetime optimization framework by estimating all the parameters simultaneously, resulting in the following objective function:

$$\underset{m_{id}, \{R, \mathbf{t}, s, m_{\text{exp}}\}_{j=1,...,N}}{\arg\min} \quad E_{feature} + w_1 E_{id} + w_2 E_{exp} + w_3 E_{exp}^s + w_4 E_{pose}^s, \tag{3.3}$$

where the first term is the $feature$ term that measures how well the reconstructed facial geometry matches the observed facial features across the entire sequence. The second and third terms are the *prior* terms used for regularizing the identity and expression parameters, which are formulated as multivariate Gaussians. The fourth and fifth terms are the

smoothness terms that penalize sudden changes of expressions and poses over time. In all of our experiments, w_1, w_2, w_3 and w_4 are set to 0.05, 0.005, 0.05, and 50 respectively.

The $feature$ term utilizes both the locations of 2D facial features from facial feature detection and the 3D depth values of the reconstructed facial features obtained from non-rigid structure from motion, resulting in the following objective $E_{feature}$.

$$E_{feature} = E_{2d} + w_d E_{depth}, \tag{3.4}$$

where the first and second terms evaluate how well the reconstructed facial geometry matches the observed 2D facial features and the 3D depth values of the reconstructed facial features. The weight w_d is experimentally set to 0.01.

3.4.2 Optimization

The direct estimate of large-scale facial geometry described in Equation 3.3 is challenging because it requires solving a complex nonlinear optimization problem with a huge number of unknowns. We develop a keyframe based optimization algorithm to address this challenge. Briefly, we first automatically select a number of key frames to represent large-scale facial geometry across the entire sequence. The extracted key frames allow us to divide the whole sequence into multiple subsequences. We then simultaneously estimate all the unknown parameters associated with all the key frames. Lastly, for each subsequence, we keep the identity fixed and compute the unknowns across the entire subsequence using the parameters reconstructed by keyframe optimization.

Keyframe extraction. Given 3D feature locations obtained from the non-rigid structure from motion, we aim to select a minimum set of frames in such a way that 3D feature locations across the entire sequence can be accurately interpolated by 3D feature locations defined at key frames. We perform principle component analysis (PCA) on 3D face shapes of the original sequence and obtain a PCA subspace for accurately representing

Algorithm 2 Keyframe Extraction

Input: the N face shapes in a sequence $V = \{v_1, ..., v_N\}$,and a tolerance threshold ϵ
Output: indices of the minimum key frames K

set $K = \{1, ..., N\}$ //initialized as the full set of video frames
while 1 **do**
  for $i = 1, ..., |K|$ **do**
    evaluate the subspace angle $SA(V_{\{K\}-i}, V)$
  end for
  $j = \underset{i}{\operatorname{argmin}}\, SA(V_{\{K\}-i}, V)$
  $\text{minError} = SA(V_{\{k\}-j}, V)$
  if $\text{minError} < \epsilon$ **then**
    $K = \{K\} - j$
  else
    break;
  end if
end while
return K

3D face shapes at any frame. We adopt a greedy strategy to find the optimal solution, initializing the key frame list with all frames in the original sequence and then incrementally decreasing it by one until the difference between the original PCA subspace and the new PCA subspace spanned by the remaining frames exceeds a user-specified threshold ϵ. The difference between the two PCA subspaces is measured by principal angle [50] and is experimentally set to 0.4. The details of our analysis algorithm are described in Algorithm 2.

Keyframe reconstruction. Since the number of the key frames is usually small (11–16 in our experiments), we can estimate all the unknown parameters at key frames using the objective function described in Equation 3.3. This can be efficiently solved by coordinate-descent optimization techniques. Note that we initialize poses and camera parameters at key frames using the reconstruction results obtained from the non-rigid structure from motion.

Keyframe interpolation. This step uses the reconstructed identity parameter as well as the recovered parameters at key frames to compute the unknown parameters at intermediate frames. This can be formulated as a keyframe interpolation problem. Specifically, given the reconstructed parameters at the starting and ending key frames, we estimate the parameters at inbetween frames in such a way that it minimizes the objective function described in Equation 3.3. During reconstruction, we assume the identity parameter is known and fixed. We initialize the camera parameters and pose parameters using the results obtained from the non-rigid structure from motion. The expression parameters are initialized by tracking each subsequence in a sequential manner.

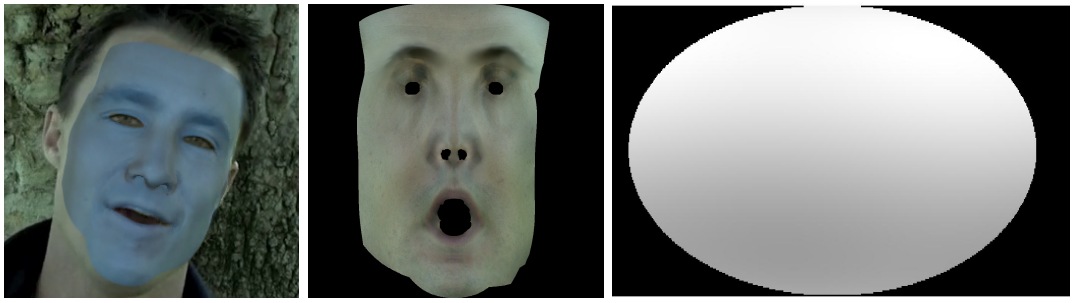
3.5 Fine-scale Detail Recovery

Dynamic facial details such as emerging or disappearing wrinkles are crucial for high-fidelity facial performance capture. This subsection describes our idea on using shading cues to add fine-scale surface details to large-scale deformation. Starting from large-scale deformation results, we compute per-pixel depth values associated with each input image as well as unknown incident lighting and face albedo by minimizing the inconsistency between the "hypothesized" and "observed" images.

3.5.1 Representation

We cast the fine-scale detail recovery problem as an image irradiance equation [51] with unknown lighting, albedo, and surface normals. We assume the face is a Lambertian surface. We further assume both lighting and face albedo are consistent throughout the whole sequence. The reflected radiance equation for a Lambertian surface is formulated as follows:

$$I(x,y) = \rho R = \rho \int \max(\mathbf{l}(\omega_i \cdot \mathbf{n}(x,y)), 0) d\omega_i, \tag{3.5}$$



(a) (b) (c)

Figure 3.2: Reconstruction of detailed facial geometry, lighting and albedo. (a) the reconstructed facial geometry overlaid on the original image; (b) and (c) show the reconstructed albedo and lighting.

where $I(x, y)$ represents the color of the pixel located at (x, y), ρ is the albedo map, and the vector **l** indicates the lighting direction and intensity, the vector **n** is the normal at the pixel located at (x, y), and ω_i represents a subtend solid angle in 3D space. R is the irradiance of the surface.

Normal map representation. The normal for each pixel is represented by spherical coordinate (θ, ϕ), *i.e.*, $n_x = \sin\theta\cos\phi$, $n_y = \cos\theta\cos\phi$, and $n_z = \sin\phi$.

Lighting and albedo. We assume that the surface of the face is Lambertian with albedo $\rho(x, y)$. The albedo is represented as RGB values in texture space. We model the light reflected by a Lambertian surface (referred to as the reflectance function) using spherical harmonics [52]:

$$R(x, y) \approx \sum_{i=0}^{N} \sum_{j=-i}^{i} l_{ij} \alpha_i Y_{ij}(x, y), \tag{3.6}$$

where l_{ij} are the lighting coefficients of the harmonic expansion, α_i are the factors that depend only on the order i, and $Y_{ij}(x, y)$ are the surface spherical harmonic functions evaluated at the surface normal. In practice, α_i in the equation can often be omitted and

the reflection function becomes

$$R(x,y) \approx \mathbf{l}^T Y(\mathbf{n}(x,y)), \tag{3.7}$$

with

$$\begin{aligned} &Y\left(\mathbf{n}\left(\mathrm{x},\mathrm{y}\right)\right) = (1, n_x, n_y, n_z, n_x n_y, n_x n_z, n_y n_z, \\ &n_x^2 - n_y^2, 3n_z^2 - 1)^T, \end{aligned} \tag{3.8}$$

where n_x, n_y, n_z are the components of the surface normals $\mathbf{n}$.

We now can synthesize an image based on the "hypothesized" lighting coefficients $\mathbf{l}$, albedo map $\rho(x,y)$ and a per-pixel normal estimate $\mathbf{n}(x,y)$:

$$I(x,y) = \mathbf{l}^T \rho(x,y) Y(\mathbf{n}(x,y)). \tag{3.9}$$

3.5.2 Objective Function

We adopt an analysis-by-synthesis strategy to reconstruct fine-scale facial geometry. Specifically, we reconstruct optimal normal maps $\mathbf{n}_i(x,y), i = 1, ..., N$ representing the facial details, as well as unknown lighting coefficients $\mathbf{l}$ and albedo $\rho(x,y)$ so that the "hypothesized" image best matches the "observed" image. We formulate the problem as an optimization problem, resulting in the following objective function:

$$\underset{\rho,\mathbf{l},\{\mathbf{n}_i\}_{i=1,...,N}}{\arg\min} \quad w_1 E_{data} + w_2 E_{albedo} + w_3 E_{reg} + w_4 E_{integrability}. \tag{3.10}$$

In all of our experiments, the weights w_1, w_2, w_3 and w_4 are set to 1, 10, 15, and 50 respectively. Note that we assume the lighting and albedo are consistent across the entire sequence. Therefore, we can combine shading cues throughout the whole sequence to model the unknown lighting coefficients and albedo.

The *data fitting* term, E_{data}, measures the inconsistency between the rendered and

observed images at each frame:

$$E_{data} = \sum_{i=1}^{N} \left\| I_i - \mathbf{l}^T \rho Y(\mathbf{n}_i(x,y)) \right\|^2. \tag{3.11}$$

Similar to Kemelmacher-Shlizerman and Basri [1], we include the two regularization terms to reduce the reconstruction ambiguity for the albedo and normal maps, resulting in the second and third terms of the objective function:

$$\begin{aligned} E_{albedo} &= \left\| \nabla^2 G(\rho) - \nabla^2 G(\rho_{ref}) \right\|^2, \\ E_{reg} &= \left\| \nabla^2 G(\mathbf{n}_i) - \nabla^2 G(\mathbf{n}_{i,ref}) \right\|^2, \end{aligned} \tag{3.12}$$

where E_{albedo} and E_{reg} are the prior terms that are used to constrain the reconstructed albedo and normal maps. The operator $\nabla^2 G(x,y) = \frac{1}{2\pi\sigma^4}(2 - \frac{x^2+y^2}{\sigma^2}) e^{-\frac{x^2+y^2}{2\sigma^2}}$ represents the Laplacian of Gaussian filter with standard deviation σ. In our experiment, we set a window size 3 for the filter with σ equals 0.5. A texture map provided by FaceWarehouse [49] is used as the reference albedo map ρ_{ref} and the reference normal map $\mathbf{n}_{i,ref}$ is initialized by normal maps of the reconstructed large-scale facial geometry at each frame.

The *integrability* term, $E_{integrability}$, is the integrability constraint described in Horn and Brooks [53] to ensure that the reconstructed normals can generate an integrable surface. Integrability is a fundamental mathematical property of smooth (C^2) surfaces. It restricts the independence of the surface normal so that

$$\frac{\partial}{\partial y}(\frac{n_x}{n_z}) = \frac{\partial}{\partial x}(\frac{n_y}{n_z}). \tag{3.13}$$

In our implementation, this constraint is formulated as:

$$E_{integrability} = \left\| \frac{\partial}{\partial y}(\frac{n_x}{n_z}) - \frac{\partial}{\partial x}(\frac{n_y}{n_z}) \right\|^2. \tag{3.14}$$

Depth recovery. We now discuss how to estimate the depth from the reconstructed normal map. We represent depth information on image space. Given a pixel location (x, y), the depth value of its corresponding surface point is represented as $z(x, y)$. The surface normal $\mathbf{n}(x, y) = [n_x, n_y, n_z]^T$ can be obtained from the depth values as follows:

$$\mathbf{n}(x, y) = \frac{1}{\sqrt{p^2 + q^2 + 1}}(p, q, -1)^T, \tag{3.15}$$

where $p(x, y) = \partial z/\partial x$ and $q(x, y) = \partial z/\partial y$.

We approximate $p(x, y)$ and $q(x, y)$ using forward differences by

$$\begin{aligned} p(x, y) &= z(x + 1, y) - z(x, y), \\ q(x, y) &= z(x, y + 1) - z(x, y). \end{aligned} \tag{3.16}$$

Combining Equation 3.15 and 3.16, we obtain the following linear constraints:

$$\begin{aligned} n_z z(x + 1, y) - n_z z(x, y) &= n_x, \\ n_z z(x, y + 1) - n_z z(x, y) &= n_y. \end{aligned} \tag{3.17}$$

Once the normal map is estimated, we can compute the corresponding depth estimate by solving the following least-squares fitting problem:

$$\arg\min_{z_i} E_{normal} + w_{d1} E_{depth1} + w_{d2} E_{depth2}. \tag{3.18}$$

The first term, E_{normal}, evaluates how well the reconstructed depth map matches the estimated normal map. We define the first term based on linear equations described in Equation 3.17.

The second term, E_{depth1}, is a Laplacian regularization term that preserves the geometric details in the reference mesh obtained from large-scale facial geometry reconstruction.

We have

$$E_{depth1} = \|\nabla^2 G(z_i) - \nabla^2 G(z_{i,ref})\|^2. \tag{3.19}$$

The last term, E_{depth2}, prevents the estimated depth z_i from being away from the reference depth $z_{i,ref}$. We have

$$E_{depth2} = \|z_i - z_{i,ref}\|^2. \tag{3.20}$$

We introduce this term because the reference depth maps, which are initialized by results obtained by large-scale deformation reconstruction, are already close to actual facial geometry. This term is critical to estimating the absolute depth values of each pixel because neither the normal fitting term (E_{normal}) nor the Laplacian term E_{depth1} can constrain the absolute depth values. Besides, this term also provides the boundary constraints for the reconstructed depth maps. In our experiments, we set the weights of boundary pixels to a higher value ("10") in order to stabilize the boundary pixels.

3.5.3 Fine-scale Geometry Optimization

Similar to large-scale deformation reconstruction, we adopt keyframe based optimization for reconstructing fine-scale facial geometry. Specifically, we first estimate the lighting coefficients, albedo map, and depth maps at key frames by solving the optimization problem described Section 3.5.2. We then use the estimated lighting coefficients and albedo map to estimate the depth maps for the rest of the frames.

We use shading cues of all the key frames to estimate the unknown lighting coefficients $\mathbf{l}$ and albedo map ρ. We initialize the normal maps using results obtained from large-scale geometry reconstruction. The albedo map is initialized by the reference albedo map.

- Step 1: we estimate the spherical harmonic coefficients $\mathbf{l}$ by finding the coefficients that best fit the current albedo and the current normal maps at all the key frames. This

requires solving a highly over-constrained linear least-squares optimization with only nine or four unknowns (see the objective function defined in Equation 3.11), which can be solved simply using least-squares techniques.

- Step 2: we update the albedo map $\rho(x, y)$ in texture space based on the estimated lighting coefficients $\mathbf{l}$ and the current normal map. This requires optimizing an objective function including the two terms (E_{data} and E_{albedo}). The objective function contains a linear set of equations, in which the first set determines the albedo values, and the second set smooths these values and can be optimized using linear system solvers.

- Step 3: we solve for normal maps by using the estimated lighting coefficients $\mathbf{l}$ and the updated albedo map $\rho(x, y)$. This requires solving nonlinear optimization described in Equation 3.10 except that we drop off the regularization term E_{albedo}. We analytically evaluate the Jacobian terms of the objective function and run a gradient-based optimization with the Levenberg-Marquardt algorithm [54].

- Step 4: we solve for depth estimates at key frames by using the estimated normal maps. This again requires solving a least-squares fitting problem described in Equation 3.18.

We repeat the procedure (Step 1, 2, and 3) iteratively, although in our experiments two iterations seem to suffice. Note that the number of degrees of freedom for lighting coefficients $\mathbf{l}$ can be either 4 or 9. We found similar results could be obtained by using the first order and second order harmonic approximations, while the first order approximation was more efficient. We thus use the first order approximation for light representation (*i.e.*, the length of $\mathbf{l}$ is 4).

Given the estimated lighting coefficients $\mathbf{l}$ and albedo ρ, the normal maps for the rest of

the sequence are estimated by solving the objective function described in Equation 3.10, except that we drop off the prior term for albedo map (*i.e.*, E_{albedo}) and the optimization is done for each single frame. Similar to Step 4, we apply least-squares techniques to recover the depth maps from the reconstructed normal maps for the rest of the sequence. Figure 3.2 shows the reconstructed geometry, albedo, and lighting for one frame of a test sequence.

3.6 Iterative Shape Refinement

Large-scale facial geometry reconstructed from high-level facial features often does not accurately match actual facial geometry because of the lack of facial features in some regions such as cheeks. We address the issue by iteratively refining large-scale facial geometry using the per-pixel normal maps obtained from fine-scale detail recovery process. In addition, we can further improve the accuracy of normal maps by using refined large-scale facial geometry. The two steps are repeated for a few times (3 in our experiment) to output the final reconstruction result. In the following, we focus the discussion on the first step as the second step is the same as fine-scale detail recovery process described in Section 3.5.

To refine large-scale facial geometry using per-pixel normal maps, we include an extra term, $normalfitting$ term, into the objective function described in Equation 3.3. The $normalfitting$ term evaluates how well the normals of the refined large-scale geometry match the normal maps obtained from fine-scale detail recovery process. This allows us to refine large-scale facial geometry for the whole face region, especially the parts without salient features such as cheeks. In practice, fine-scale facial details such as wrinkles often dominate the normal fitting process because of large normal residuals. To address the issue, we filter the normal maps obtained from fine-scale detail recovery process by applying an exponential function to adjust the normal differences so that small differences

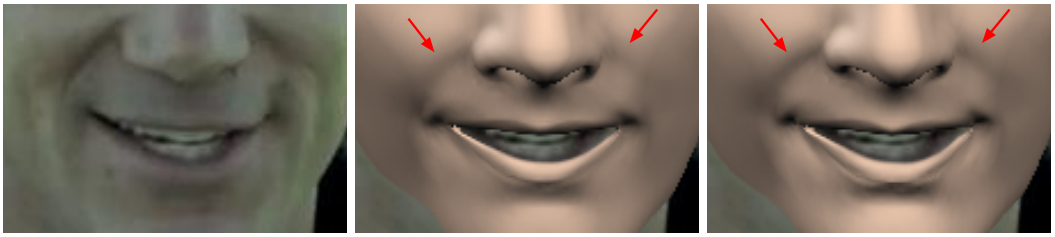


Figure 3.3: Large-scale facial geometry refinement: (left) the input image; (middle) large-scale facial geometry before the refinement; (right) large-scale facial geometry after the refinement. Note that the nasolabial folds are lifted up with the constraint of fine-scale normal map.

are preserved and large differences are marginalized. In our implementation, we solve large-scale geometry refinement using Levenberg Marquardt optimization [54]. Note that we keep 3D head poses fixed during the optimization and only the identity and expression weights are refined. Figure 3.3 shows a side-by-side comparison between the original and refined large-scale facial geometry.

Because depth maps estimated from each frame are view-dependent, fine-scale facial details invisible to the camera are missing. The resulting depth maps are also difficult to be integrated with large-scale facial geometry for facial rendering and manipulation. To address the challenges, we bake fine-scale details (*i.e.*, the difference between the estimated depth map and large-scale facial geometry) into a displacement map. Briefly, we describe each depth pixel as 3D points in a global coordinate system and project them onto the texture space of large-scale face mesh. Note that the texture map and the texture coordinates of large-scale facial geometry are defined in advance by the artist. We generate the displacement map by computing 3D offsets between the depth points and their corresponding points on the large-scale facial geometry in the same texture space. We automatically fill in missing displacement values by using Poisson image editing technique [55]. Therefore, we can render the reconstructed facial performances using large-scale face mesh and its

Figure 3.4: High-fidelity facial performance capture: from left to right, we show the input image data, the reconstructed head poses and large-scale facial geometry, the reconstructed high-fidelity facial geometry overlaid on the original image data, and the reconstructed high-fidelity facial data.

displacement map using GPU accelerated displacement mapping techniques [56].

3.7 Results and Applications

In this subsection, we first demonstrate the power and effectiveness of our system by capturing a wide range of high-fidelity facial performances using our proposed system (Section 3.7.1). We show our system achieves better results on video-based facial capture by comparing against alternative systems (Section 3.7.2). We quantitatively evaluate the performance of our system on synthetic image data generated by high-fidelity 3D facial performance data captured by Huang and his colleagues [29] (Section 3.7.3). Finally, we show novel applications of our reconstructed facial data in facial video editing (Section 3.7.4).

3.7.1 Test on Real Data

We evaluate the performance of our system on video sequences of four different subjects with lengths ranging from 344 (11s) to 846 frames (28s). Three of videos are downloaded from the Internet. All of the test videos have a resolution of 640×360. Figure 3.4 shows some sample frames of our results.

3.7.2 Comparisons

We have evaluated the effectiveness of our system by comparing against alternative techniques.

Comparison against Kemelmacher-Shlizerman and Basri [1]. Our fine-scale detail reconstruction builds on the success of modeling fine-scale facial geometry from a single image [1]. Figure 3.5 shows a side-by-side comparison between our method and their method. The reference face template required by their work is based on the neutral expression mesh model of the subject. The same reference albedo is used in both methods. As shown in Figure 3.5, our system produces more fine-scale facial details and obtain more

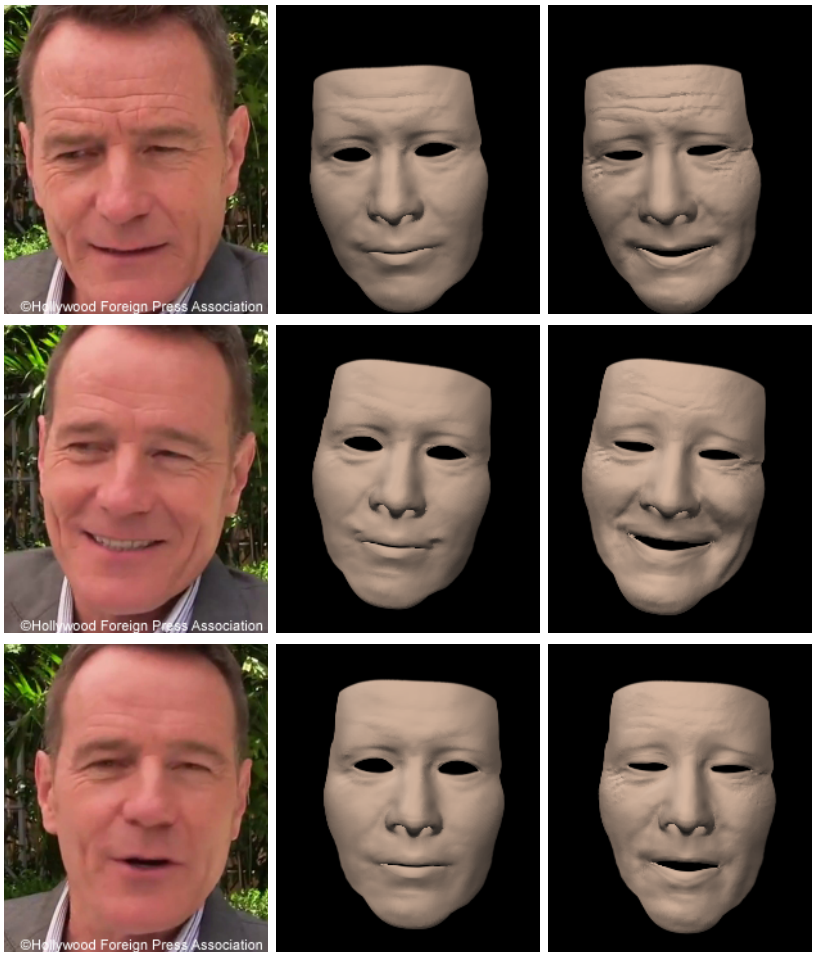


Figure 3.5: Comparison against Kemelmacher-Shlizerman and Basri's single image facial reconstruction on real data. From left to right: input images, their results, and our results.

accurate geometry than theirs.

Comparison against Garrido et al. [2]. We compare our method against state of the art in facial performance capture using monocular videos [42]. The resolution is 900×600 and the total number of frames is 538. We show a side-by-side visual comparison between the two results. As shown in Figure 3.6, our method captures richer details and produces better facial geometry than their method. Figure 3.6 also shows that our method is more robust to large pose variations and produces more accurate facial reconstruction results for

Figure 3.6: Comparisons against Garrido et al.'s facial reconstruction method on a monocular video. From left to right: input, their reconstruction results, and our results.

extreme head poses.

3.7.3 Evaluation on Synthetic Data

We evaluate the importance of key components of our system based on synthetic data generated by high-fidelity facial data captured by Huang and colleagues [29]. The whole test sequence consists of 300 frames with large variations of expressions and poses. We use ground truth head poses, facial geometry and texture to synthesize a sequence of color images. The resolution of image data is set to 640×480.

The experiment is designed to show the importance of three key components of our facial reconstruction system: including "large-scale geometry and pose reconstruction" de-

scribed in Section 3.4, "batch-based fine-scale geometry optimization" described in Section 3.5, and "iterative shape refinement" described in Section 3.6. We test four different methods on the synthetic data set. The four methods are noted as "Kemelmacher-Shlizerman and Basri", "Linear", "Batch", and "Our method" respectively. "Kemelmacher-Shlizerman and Basri" [1] initializes the template mesh using the neutral expression model of the subject and applies the least-squares algorithm to solve lighting coefficients, albedo, and facial geometry in each frame separately. Similar to "Kemelmacher-Shlizerman and Basri", "Linear" method solves facial geometry, lighting coefficients, and albedo in each frame separately. However, it improves "Kemelmacher-Shlizerman and Basri" by initializing the reference template model and poses using large-scale geometry and poses obtained from Section 3.4. "Batch" method improves "Linear" method by assuming consistent lighting and albedo across the entire sequence and solving the unknown lighting coefficients, albedo and facial geometry based on shading cues throughout the whole sequence. "Our method" further improves the "Batch" method by running iterative shape refinement described in Section 3.6.

Figure 3.7 shows average reconstruction errors for all methods. We evaluate the reconstruction accuracy by computing the average normal direction discrepancy between ground truth normal maps and normal maps reconstructed from each method. As we can see, the fitting error decreases as we gradually improve the method.

3.7.4 Applications

With detailed facial geometry, albedo, and illumination recovered from monocular video sequences, our method allows users to easily edit the underlying geometry and albedo of the face.

Albedo editing. Albedo editing allows the user to edit the albedo map in texture space. Once the albedo map is edited, we can combine it with the reconstructed lighting

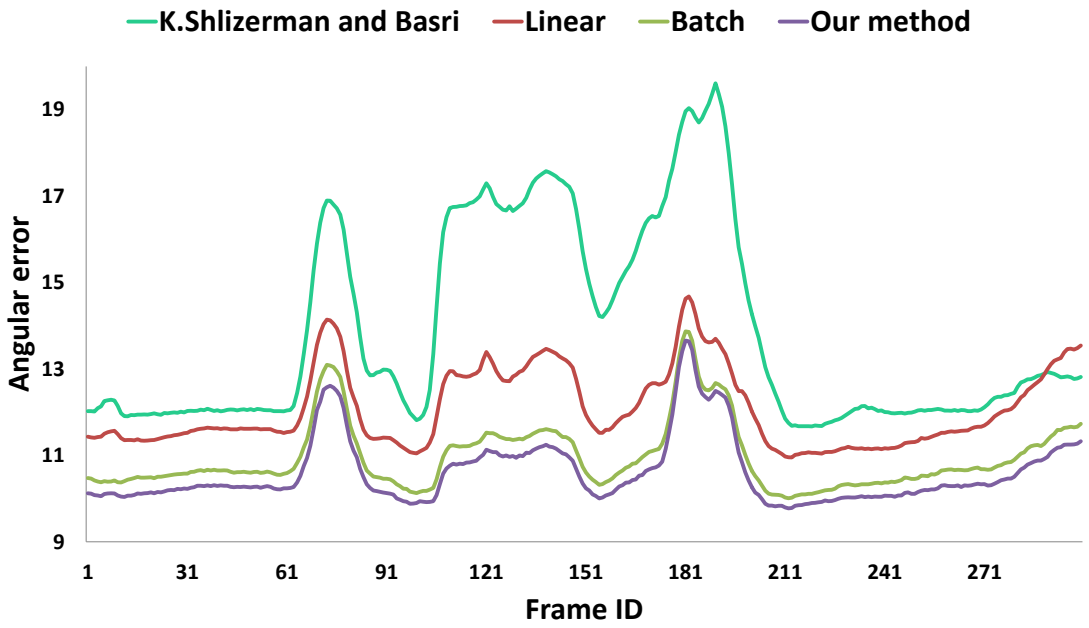


Figure 3.7: Evaluation of key components of our system on synthetic image data.

coefficients and captured facial geometry to render a new sequence of images based on the rendering equation described in Equation 3.9. We further replace the original video data with the "synthesized" video data based on a user-specified mask. By solving Poisson equations with boundary constraints at every frame, we seamlessly blend the rendered image data with the original image data. Figure 3.8 (top) shows two sample frames of albedo editing results, where we add a beard to the subject.

Large-scale facial geometry editing. We can edit the underlying large-scale facial geometry data at any frame and use the modified facial geometry to edit facial video data across the entire sequence. Figure 3.8 (bottom) shows a video editing result based on large-scale face geometry editing. In this example, we modify the underlying facial geometry of the subject under the neutral expression. We then transfer large-scale facial deformation of the subject to the new subject (*i.e.*, the edited face) via the deformation transfer technique described by Sumner et al. [57]. Specifically, for every pixel in the image, we find the

Figure 3.8: Albedo editing: adding a beard (top) and large-scale facial geometry editing (bottom). At the bottom row, the left two images show the original and edited large-scale facial geometry. The right two images show the original and edited image data.

corresponding point on the surface of the original facial geometry and project the 3D offset between the original and edited facial geometry onto the image space to obtain the corresponding image displacement. The per-pixel image displacement map is used to warp the original image into a new image (*i.e.*, the edited image). Note that the expressions of the original subject are faithfully transferred to the new facial subject because our system can accurately reconstruct a space-time coherent 3D face geometry.

Fine-scale facial editing. In Figure 3.9, we modify fine-scale facial details reconstructed from the original video sequence to remove the wrinkles from the entire sequence. For this purpose, we first smooth the displacement map of each frame with a low-pass filter. We then use the edited facial geometry, as well as the reconstructed lighting coefficients and albedo, to render a new sequence of images. At a final step, we paste the "rendered" video data back to the original video data in the same way as albedo editing. Note that the shading details caused by fine-scale geometric details are removed in the edited video se-

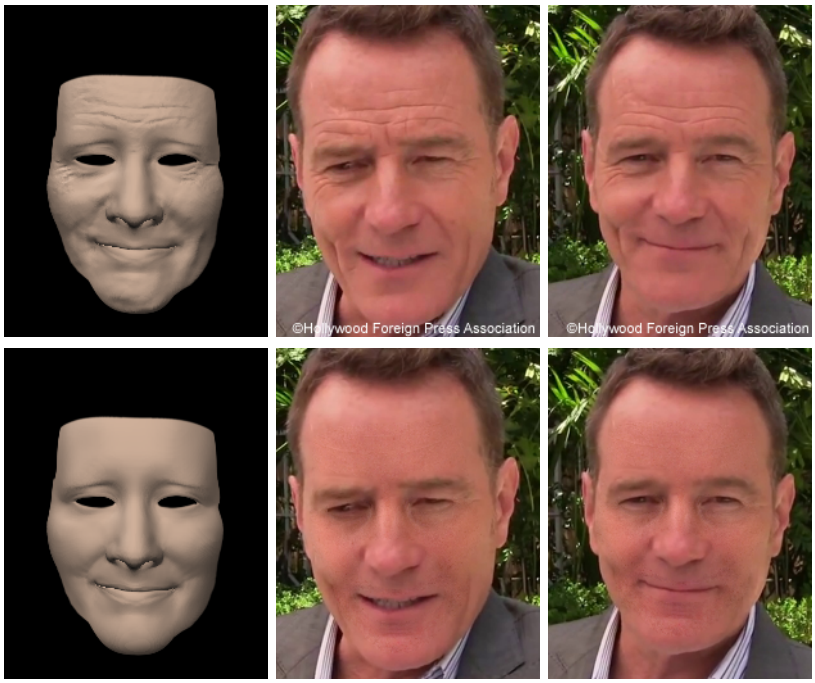


Figure 3.9: Fine-scale geometric detail editing: wrinkle removal. The first column shows the original facial geometry with fine geometric details and the edited geometry after removing fine geometric details. The second and third columns show the video editing results.

quence, while skin texture details and appearance variations corresponding to large-scale facial deformations are well preserved.

3.8 Discussion

In this section, we have developed an end-to-end system that captures high-fidelity facial performances from monocular videos. The key idea of our method is to utilize automatically detected facial features and per-pixel shading cues, along with multilinear facial models, to infer 3D head poses, large-scale facial deformation, and fine-scale facial detail across the entire sequence. Our system is appealing for facial capture because it is fully automatic, offers the lowest cost and a simplified setup, and can capture both large-scale deformation and fine-scale facial detail. We have tested our system on monocular

videos downloaded from the Internet, demonstrating its accuracy and robustness under a variety of uncontrolled lighting conditions and overcoming significant shape differences across individuals. We have explored novel applications of captured facial performance data in facial video editing, including removing wrinkles, adding a beard, and modifying underlying facial geometry.

(a) (b) (c)

Figure 3.10: Limitations: reconstruction artifacts caused by strong cast shadows, non-Lambertian reflectance, and occlusions caused by glasses. (a) shows artifacts produced by strong cast shadows around the right eye, the nostril and the lip region; (b) shows artifacts due to non-Lambertian specular highlight on the forehead; (c) shows the reconstruction result under significant occlusions caused by sun glasses.

The current system has a few limitations. First, it ignores cast shadows, which can be created by the non-convex shapes on the face. Figure 3.10 (a) illustrates such a concern. Fine-scale geometry is overfitted around the right eye, the nostril, and the lip region where

strong cast shadows occur. Additionally, the current system assumes the face has Lambertian reflectance. Human faces, however, are not exactly Lambertian since specularities can be observed in certain face regions. For example, the specular highlight on forehead (Figure 3.10 (b)) produces an incorrect ridge on the reconstructed facial geometry. Finally, our system cannot distinguish occlusions caused by facial hair, glasses, hands or other objects, *e.g.*, glasses shown in Figure 3.10 (c). Though fine-scale details are still recovered, artifacts are introduced due to the occlusions caused by sunglasses. In the future, we would like to explore how to integrate shadow cues into the reconstruction framework and how to extend the current framework to handle non-Lambertian facial reflectance.

The current system runs offline and requires the whole video sequence for reconstruction. Thus, it cannot be applied to realtime applications such as 3D facial performance capture on a live video stream. Meanwhile, the movements of other facial components such as eye gaze and tongue are missing. This is also a major limitation for most of the current facial performance capture systems [58, 59, 60]. In next section, we refine the 2D facial feature tracking and large-scale facial deformation reconstruction component to make it realtime, and complement it with a novel realtime eye gaze tracker to simultaneously track 3D head pose, facial expression, and 3D eye motion as well.

4. REALTIME 3D FACIAL PERFORMANCE AND EYE GAZE ANIMATION USING A SINGLE RGB CAMERA*

In this section, we present a realtime 3D facial performance and eye gaze capture system that simultaneously captures the coordinated movements of 3D eye gaze, head poses, and facial expression deformation using a single RGB camera. We first refine the robustness and efficiency large-scale facial reconstruction by introducing a local binary feature (LBF) based facial feature detection/tracking method. A novel convex-hull based feature selection constraint and a robust initialization method for tracking are proposed for a better robustness. We then introduce a convolutional neural network based pose and shape regression that reconstructs 3D head pose and expression as well as 2D facial features directly from the input face image. Finally, we complement the 3D facial performance capture component with an efficient 3D eye gaze tracker, which automatically and robustly tracks the 3D eye gaze in the Maximum A Posterior (MAP) framework.

Our final performance capture system is robust and fully automatic, allowing for simultaneous capture of 3D head poses, large-scale facial deformation, and 3D eye gaze in realtime using a single RGB camera. We have tested our system on both live video streams and Internet videos, demonstrating its accuracy and robustness under a variety of uncontrolled lighting conditions and overcoming significant differences of races, genders, shapes, poses, and expressions across individuals.

4.1 Background

Our realtime facial performance system automatically tracks 3D eye gaze, 3D head poses, and facial expression deformation using a monocular RGB camera. Therefore, we

*Part of this section is reprinted with permission from “Realtime 3D eye gaze animation using a single RGB camera" by C. Wang, F. Shi, S. Xia, and J. Chai, *ACM Transactions on Graphics*, vol. 35, no. 4, p. 118, 2016. DOI: 10.1145/2897824.2925947. Copyright 2016 by ACM, Inc.

focus our discussion on methods and systems developed for acquiring 3D facial performances and gaze motion.

4.1.1 Facial Performance Capture

Facial performance capture has a long history in computer graphics and vision. Various methods have been proposed in film and game production such as marker-based motion capture systems [27, 28, 29], marker-less facial capture that uses depth and/or color data obtained from structured light systems [30, 31, 33], and multi-view stereo reconstruction systems using RGB images from multiple cameras [34, 35, 36, 37]. Recent advance in 3D depth sensing has enabled a number of facial performance capture techniques using RGBD cameras [44, 45, 46, 61, 47]. Notably, Weise and colleagues [44] used RGBD image data to construct a user-specific DEM in preprocessing and track facial expressions at runtime. Most recently, Bouaziz and colleagues [45] and Li and colleagues [46] concurrently developed realtime monocular face trackers based on a runtime shape correction strategy for combined color and depth data.

A more appealing solution for facial capture is to use a monocular RGB camera as it offers the lowest cost and a simplified setup. These methods first locate facial landmarks such as the nose tip and then use them to drive the 3D facial animation. Recent advances for locating/tracking facial landmarks include constrained local model [62, 63] and boosted regression [26, 64, 3]. In particular, Cao and his colleagues [26] proposed an explicit two-level cascaded shape regressor for facial feature detection and Ren and his colleagues [3] further refined the accuracy and efficiency of facial feature detector by utilizing the locality principle and a local binary feature based shape regressor.

Cao and colleagues [40] extended the idea of cascaded shape regression [26] for 3D facial capture. They trained a user-specific 3D shape regressor and then use it to track 3D shape from 2D image sequences at runtime. Recently, they proposed a user-independent

displacement dynamic expression regression that adaptively refines the camera matrix and user identity during tracking [58]. Most recently, they further extended the idea to realtime high-fidelity facial capture by adding a local user-specific detail regressor [59]. Besides these online techniques, there are also offline systems that apply shape-from-shading to capture detailed and dynamic 3D facial geometry [2].

Our convolutional neural network (CNN) based pose and shape regression is relevant to methods on CNN-based face alignment and reconstruction. Sun et al. [65] and Fan and Zhou [66] use CNNs for 2D landmark regression. However, they do not reconstruct 3D head pose or facial deformation. Most recently, Jourabloo and Liu [67] propose a cascaded CNN regressor for face alignment through the 3D morphable model fitting. The shape (identity and expression) and camera parameters are incrementally refined through a set of CNNs. They extract patches around each facial landmark and compose them into a single image as the input for CNN. Zhu et al. [68] propose to combine both RGB and depth images for CNN regression. Our method is different from theirs. First, we propose a novel 2D canonical texture space to represent images as the CNN input. The proposed image space effectively represents the accurateness of the current shape estimation and does not contain any discontinuities such as the boundaries of the patch composition [67]. Second, our method assumes the identity and camera intrinsics (such as focal length and optical center) are known and regresses pose and expression for each frame. Thus, it can be directly applied to video tracking. In contrast, their methods estimate identity and camera parameters for each input image, which could bring inconsistency for video tracking. Finally, we add an additional 2D displacement regression network that further increases the accuracy of the 2D facial feature locations.

This section enhances facial performance capture by adding a realtime 3D eye gaze tracker into the facial capture. This enhancement allows us to capture coordinated movements between 3D head poses, facial expression, and eye gaze. It is worth mentioning

that our framework is flexible and our eye gaze tracker can be integrated with any 3D facial capture system using RGB images, *e.g.*, [36, 37, 45, 46, 59], to capture coordinated movements between 3D head poses, facial deformation, and eye gaze.

4.1.2 Eye Gaze Tracking

Eye gaze tracking and eye detection have been an active research topic in the field of human-computer interaction and computer vision for many decades. Previous methods for eye detection and eye gaze tracking are mainly focused on 2D gaze detection and tracking and can be classified into two categories: IR-illumination based approaches and image-based approaches. The active IR illumination based approaches exploit the spectral (reflective) properties called cornea reflection under IR illumination to efficiently detect iris and pupil pixels, while the image-based approaches aim to detect or track eye gaze based on the shapes and/or appearances of the human eyes.

Active IR based methods (*e.g.*, [69]) is one of the most successful approach for eye gaze capture. Due to the simplicity and effectiveness of the method, almost all the commercial eye trackers are based on this technique. However, the IR based method is intrusive because it requires users to wear a special glass or set up a dedicated IR device for gaze capture. Additionally, unlike our method, it is often focused on 2D eye gaze capture alone and is also not flexible for capturing 3D eye gaze in uncontrolled videos.

Traditional methods in image-based approaches can be further divided into three categories: template matching [70], appearance based methods [71, 72], and feature based methods [73, 74]. However, those methods are often not robust enough to handle variations in lighting, subjects, head poses, and facial expressions. Recent efforts are mainly focused on applying cascade shape regression [26] or deep learning methods for pupil center detection. By adding two extra landmarks around the eye pupil center, those methods can detect both facial feature locations and the center of the pupil in a single image. Our

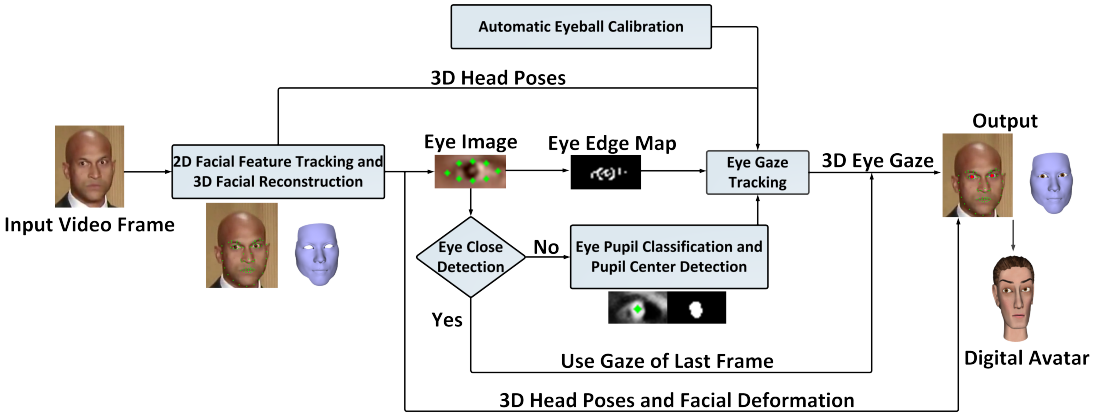


Figure 4.1: An overview of our realtime 3D facial performance and eye gaze capture system.

goal is different from theirs because we focus on 3D facial performance capture with 3D eye gaze rather than 2D facial feature detection and eye detection.

4.2 Overview

We aim to build a realtime facial performance capture system that robustly and accurately tracks 3D eye gaze motion using a monocular RGB camera. The problem is challenging because head poses, facial expression deformation and eye gaze motion are often coupled together. An accurate estimate of 3D eye gaze often requires an accurate estimate of 3D head poses and facial deformation around both eyes. In addition, ambiguity caused by the loss of depth information in the projection from 3D to 2D and unknown camera parameters and lighting conditions further complicate the problem. To address the challenges, we propose an end-to-end facial performance system that simultaneously tracks 3D head poses, eye gaze and facial expression deformation using a single RGB camera. We summarize the main components as follows (see Figure 4.1).

The first component is 2D facial feature detection and 3D facial performance reconstruction. We start the process by automatically detecting and tracking important facial

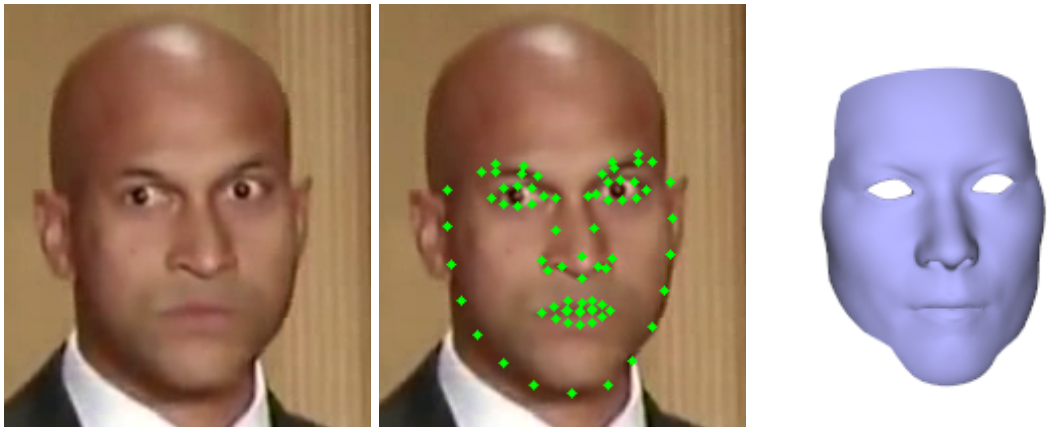
features such as the nose tip in monocular video sequences. We introduce a data-driven 3D facial reconstruction technique to reconstruct 3D head poses and large-scale expression deformation using multilinear expression deformation models. As an alternative, a convolutional neural network based pose and shape regression framework is also proposed.

The second component is pixel classification for iris and pupil and pupil center detection. We introduce a novel user-independent pixel classifier to automatically annotate iris and pupil pixels in the eye region, which is bounded by detected facial landmarks in the eye region. We further obtain the 2D location of the pupil center by applying the mean-shift algorithm [21] to the classified iris and pupil pixels. We discuss how to extract the outer contour of iris *i.e., limbus* to further improve the robustness and accuracy of our gaze tracker.

The third component is automatic eyeball calibration. Tracking 3D eye gaze across an entire video sequence requires not only modeling the geometry of 3D eyeball but also estimating the location of the eyeball and the size of the iris and pupil region. We approximate the geometry of the eyeball with a sphere of a particular radius (12.5mm), which corresponds to the average radius of adult eyeballs. We introduce an eyeball calibration step, which is automatically done at the beginning of each capture, to estimate the 3D location of the eyeball and the size of the iris and pupil region.

The fourth component is eye gaze tracking. We represent the state of 3D eye gaze based on the location of each pupil center on the surface of the eyeball sphere. We sequentially update the state of the eye gaze based on the detected 2D pupil center, the outer contour of iris *i.e., limbus*, and estimated 3D head poses. We formulate the problem in the Maximum A Posteriori (MAP) framework and apply importance sampling to infer the most probable state of eye gaze.

The last component is eye close detection. In practice, we have observed that the eye gaze tracking algorithm often fails when eye blinks. This leads us to introduce a novel eye



(a) (b) (c)

Figure 4.2: 2D facial feature tracking and 3D face reconstruction. (a) the input image; (b) tracked 2D facial features; (c) reconstructed 3D pose and facial deformation without eye gaze motion.

close detector to automatically detect whether the eye is open or closed. Once the eye is closed, we disable the iris and pupil pixel classifier and gaze tracking and choose to predict the state of the eye gaze using the results from the previous frame. In addition, we discuss how to use eye gaze constraints embedded in training data to further improve the accuracy and robustness of the system.

Note that the eye gaze capture part is done by our colleague Congyi Wang at Institute of Computing Technology (Chinese Academy of Sciences). So we only briefly introduce the eye gaze tracking part here and will describe our methods on realtime facial performance reconstruction in detail in the following subsections.

4.3 2D Facial Feature Tracking and 3D Face Reconstruction

This subsection discusses how to reconstruct 3D head poses and facial expression deformation from an RGB video sequence which are critical to the 3D gaze tracker. We start with 2D facial features detection/tracking (Figure 4.2 (b)) which builds on local binary feature (LBF) based regression [3]. We then reconstruct the 3D head poses and deformation

(Figure 4.2 (c)) using the tracked 2D features.

4.3.1 2D Facial Feature Detection/Tracking

This step aims to detect and track 2D facial features from a monocular video. This task is achieved by a local binary feature (LBF) based regression, which has been shown to outperform the cascaded regressors [26] in both accuracy and efficiency. Local binary feature is a long 1D vector assembled by 1D binary features obtained at each landmark. Given this vector, the facial shape S (*i.e.*, a collection of 2D facial feature locations) is progressively refined by estimating a shape increment ΔS stage-by-stage. The shape increment ΔS^t at stage t is regressed using the input image and extracted local binary features using random forests (Equation 4.1):

$$\Delta S^t = W^t \Phi^t(I, S^{t-1}), \tag{4.1}$$

where W is a linear regression matrix, I is the input image, S^{t-1} is the shape at the previous stage, and Φ^t is the mapping function (random forests) which maps I and S^{t-1} to the local binary features. Specifically, $\Phi = \{\phi_1, ..., \phi_n\}$, where each ϕ is a random forest for a particular facial landmark, and n is the number of facial features.

The local binary features are obtained in two steps. Given an image and the current 2D shape, the first step is to use each local mapping function ϕ_i to extract a local binary vector for feature i (Figure 4.3, first row). Then, all n local binary features are concatenated together to form the final local binary feature vector (Figure 4.3, second row).

Though the LBF regressor in Ren et al. [3] is fast and robust, we found the result could degenerate significantly for cases with extreme poses and low-quality images. We have made two refinements in the training/prediction process.

First, we introduce a convex-hull based feature selection scheme that effectively avoids selecting pixels at the background in the random forest training process. To train a ran-

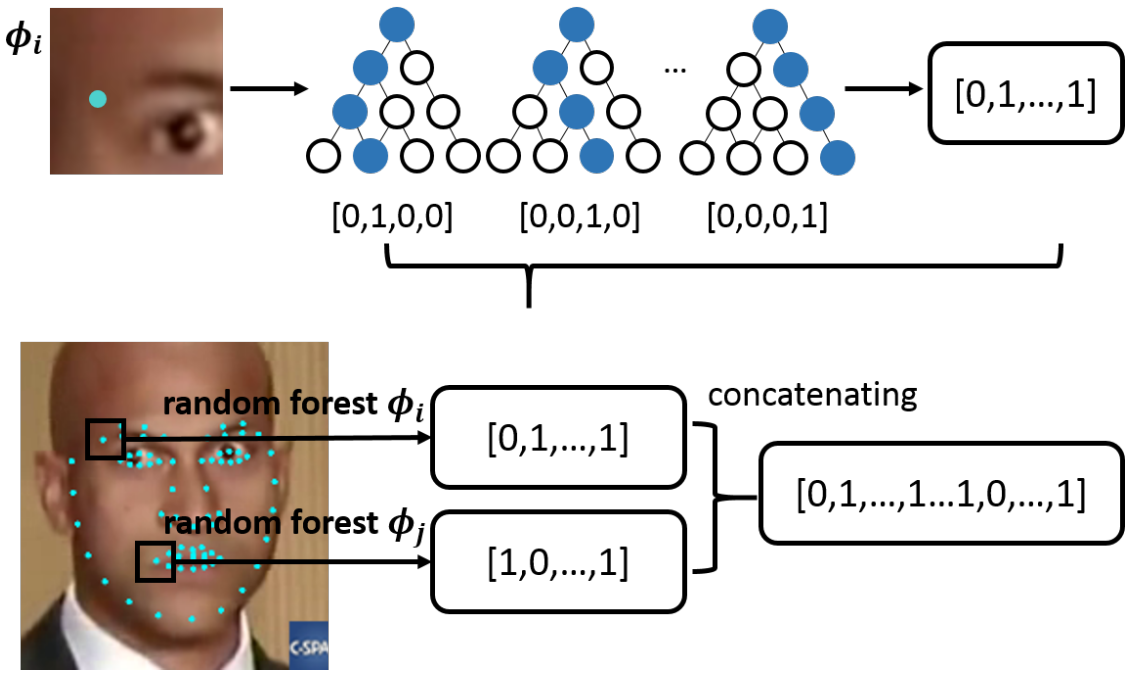
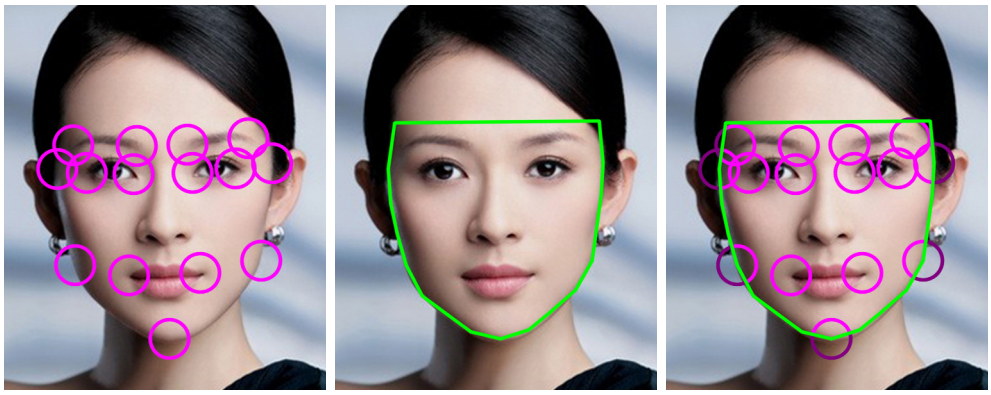


Figure 4.3: The local binary features. Top row: random forest is used as local mapping function for each feature and outputs a vector as local binary feature; bottom row: the local binary feature for each facial feature is concatenated together as the final local binary features.

dom forest for a particular landmark, we need to select most discriminative feature pairs (giving rise to maximum variance reduction) for each node from a set of candidate feature locations. These candidates are firstly sampled at a reference shape and then transferred to each of the training samples through a similarity transform. Ren et al. [3] proposed to select features at the local region of a landmark (Figure 4.4 (a)). At the beginning stage, the shape could be inaccurate and the size of the local region is set to a large value to take this into account and focus on large global shape variations. As the stage increases, the size of the local region decreases for better subtle adjustments.

However, this method might select locations that are totally outside the facial region, which could be random background. This is the case especially for the contour landmarks. As a result, the landmark locations can be inaccurate when the input images contain ran-

dom noise and/or a background that is different from those of the training images. To address this issue, we refine the feature selection process by further constraining the sample locations to be inside the convex hull of the 2D feature points (Figure 4.4). Unreliable feature locations at background can be effectively ignored. Note that this scheme is only performed on the reference shape at the training stage. Once the good feature pairs are selected, they will be directly used in the prediction. No further convex-hull check is needed at the prediction stage.



(a) (b) (c)

Figure 4.4: Convex-hull based feature selection for random forest training. (a) the original feature sampling process selects a local region within a certain radius around each landmark; (b) convex-hull of the facial features which is used to constrain the local region; (c) our feature sampling process only selects the local regions inside the convex hull, thus the unreliable regions at background are effectively excluded.

In addition, we refine the initialization step at the prediction stage for a better robustness to pose variations. In the tracking process of Ren et al. [3], the initial shape S^0 is set as the mean-shape aligned to the previous landmark locations with a similarity transform. However, our experiments indicate that this strategy might output inaccurate results for faces with extreme poses. An alternative is to use the k nearest neighbors to the previous

result as the initializations and take the mean/median of the results as the output. Though this strategy does address the issue effectively, it might output less stable results as the nearest neighbors are more sensitive to the shape changes. As a compromise, we combine these two strategies to draw benefits from both ends. Specifically, given the landmark locations from the previous frame, we find the nearest neighbors and the aligned mean shape. We then count the number of feature differences between the nearest neighbor and the aligned mean shape which are larger than a threshold (ϵ_1). If this number is no larger than a given upper bound (ϵ_2), we use the aligned mean shape as the initialization. Otherwise, the k nearest neighbors ($k = 3$) are used. Our experiment shows that this idea is quite effective to obtain both frame-frame smoothness and robustness to large pose variations. ϵ_1 and ϵ_2 are experimentally set to 15 pixels and 7 respectively.

Figure 4.5 shows the comparisons between our method and Ren et al. [3] which demonstrates the effectiveness of the two refinements.

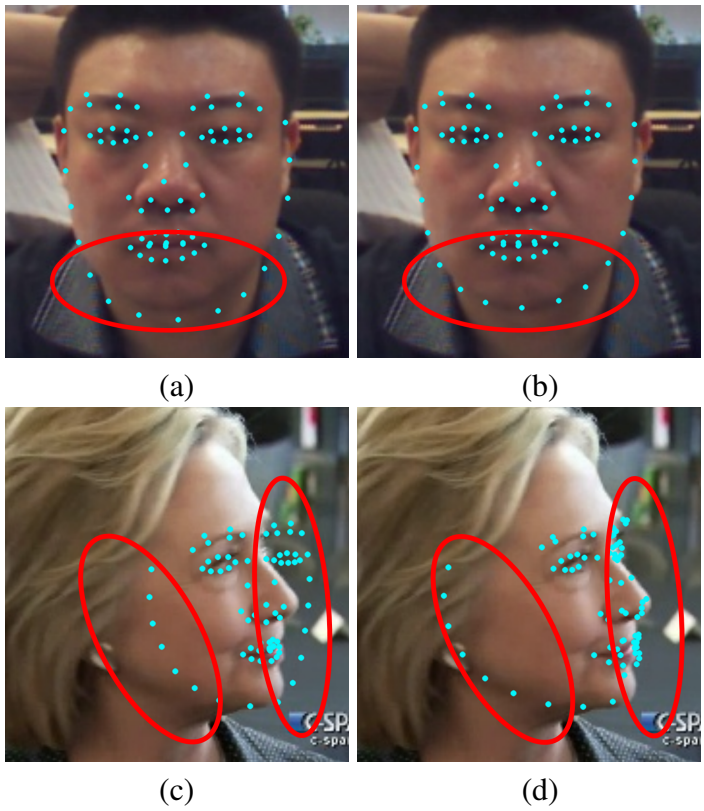
4.3.2 3D Facial Performance Reconstruction

We now describe how to reconstruct 3D facial deformations and head poses from the tracked 2D locations.

Similar to our work on high-fidelity facial reconstruction in Section 3, we represent the 3D facial models using multilinear models [75, 40]. Specifically, we describe a 3D face using two low-dimensional vectors controlling the identity and expression of the 3D face respectively:

$$M = R(C_r \times_2 m_{id}^T \times_3 m_{exp}^T) + T, \tag{4.2}$$

where M represents large-scale facial geometry of an unknown subject, R and T represent the global rotation and translation of the subject, C_r is the reduced core tensor, and m_{id} and m_{exp} are identity and expression parameters respectively. Our multilinear model was constructed from FaceWarehouse [49], which contains face meshes corresponding to 150



(a) (b)

(c) (d)

Figure 4.5: Comparison against the original local binary feature based regression proposed by Ren et al.. (a) & (c) are results from their method, and (b) & (d) are our results. The top row shows the results on an image with a low quality, and the bottom row shows the case with an extreme pose.

identities and 47 facial expressions. In our experiment, the numbers of dimensions for the identity and expression parameters are set to 50 and 25.

By assuming an ideal pinhole camera model, the projected 2D features at image space can be represented as:

$$\mathbf{p}_k = Q(R((C_r \times_2 m_{id}^T \times_3 m_{\text{exp}}^T)^{(k)}) + T), \tag{4.3}$$

where $Q = [f\ 0\ u; 0\ f\ v; 0\ 0\ 1]$ is the ideal pinhole projection matrix, (R, T) is the 3D rotation and translation, f is the focal length, and (u, v) is the principal point.

The goal here is to minimize the difference between the detected 2D features and the projected 2D features from the hypothesized face model. Similar to the large-scale reconstruction in Section 3.4, extra prior and smoothness terms for expression and pose are imposed. Note that the identity weight and focal length are only estimated at the start of the video and then fixed for the remaining frames. We follow the same binary search process in Cao et al. [40] to find the optimal focal length. The principle point (u, v) is set to the center of the image. For the rest of the frames, the objective function is as follows:

$$\underset{m_{\exp},R,T}{\arg\min} \quad E_{feature} + w_1 E_{exp} + w_2 E^s_{exp} + w_3 E^s_{pose}, \tag{4.4}$$

where the first term $E_{feature} = \sum_k \|\mathbf{p}_k - \mathbf{s}_k\|^2$ is the $feature$ term that measures how well the reconstructed facial geometry matches the observed facial features. The second term $E_{exp} = (w_{exp} - \overline{w_{exp}})^T \Sigma^{-1}_{exp} (w_{exp} - \overline{w_{exp}})$ is the *prior* term used for regularizing the expression parameters, which is formulated as a multivariate Gaussian with mean $\overline{w_{exp}}$ and covariance matrix Σ_{exp}. The third and fourth terms ($E_{exp} = \|w_{exp,t} - w_{exp,t-1}\|^2$ and $E_{pose} = \|pose_t - pose_{t-1}\|^2$) are the *smoothness* terms that penalize sudden changes of expressions and poses over time. In all of our experiments, w_1, w_2 and w_3 are set to 0.00001, 100, and 10 respectively.

The pose smoothness term constrains large rotation and translation changes between frames:

$$E^s_{pose} = w_4 E^s_{rotation} + w_5 E^s_{translation}, \tag{4.5}$$

where w_4 and w_5 are set to 1 and 0.1 respectively.

The refined 2D feature detection and large-scale performance reconstruction component runs in realtime (23ms/frame), which is tested on a PC with an Intel(R) Xeon(R) CPU 3.3GHz and 16GB RAM. Note that the current implementation is based on CPU

with no parallel optimization, and the performance can be further improved by using GPU acceleration.

4.4 Convolutional Neural Network (CNN) Based 3D Pose and Expression Regression

We have achieved realtime 2D facial feature detection and large-scale facial deformation reconstruction. However, the two-step based process has the following limitations. First, once the 2D facial feature detection/tracking fails, the 3D reconstruction step will still try to fit the bad locations and output incorrect pose and facial deformation. Second, the 3D reconstruction step only utilizes sparse 2D features as constraints and is vulnerable to the 2D-3D ambiguities.

An ideal alternative is to use the pixel information of the entire face image to directly infer pose and facial deformation. Meanwhile, recent advances [76, 65, 77] have shown that convolutional neural network (CNN) is effective to extract the intrinsic image features and correlation from the features to the output. This inspires us to present a CNN-based facial performance tracking framework. The key idea is to use a cascaded CNNs to progressively refine the pose and expression parameters as well as the 2D feature locations.

In the following, we will first explain the representation of the facial deformation and 2D facial features and then describe the proposed framework in detail.

4.4.1 Representation

We use the multilinear model (Equation 4.2) to represent the facial deformation. We represent 2D facial features by the projection of vertices and a 2D displacement vector (Equation 4.6). Note that the direct projection of pre-selected vertices can be inconsistent with the 2D feature locations. For example, the eye brows of different people can be either thin or thick, but the pre-selected 3D eye brow vertices only form a fixed size. Thus, it is necessary to complement the vertex projections with a 2D displacement vector for accurate

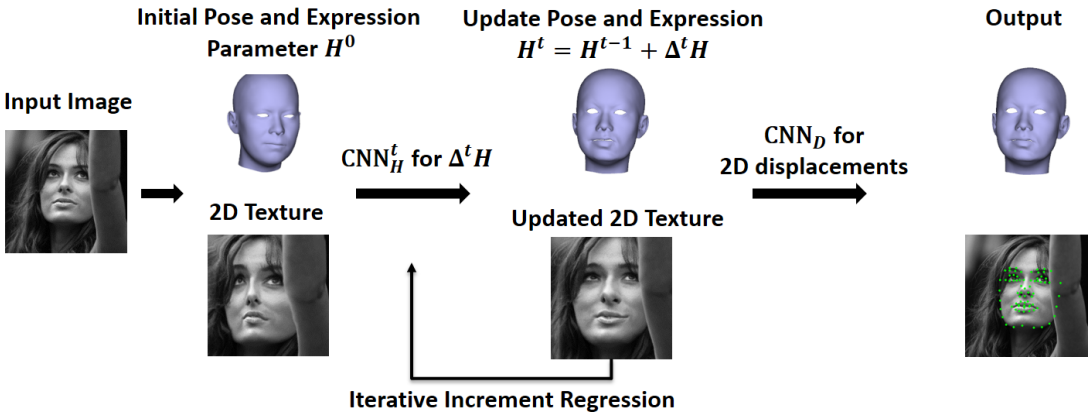


Figure 4.6: The framework of our Convolutional Neural Network (CNN)-based pose and expression regression. Given an input image and initial pose and expression parameters, we progressively refine the pose and expression parameter using the CNN cascades. At a final step, an additional CNN is used for regressing 2D displacements which are added to the projected vertex locations to get the final 2D locations.

landmark locations.

$$\mathbf{p}_k = Q(R((C_r \times_2 m_{id}^T \times_3 m_{\text{exp}}^T)^{(k)}) + T) + d_k, \tag{4.6}$$

where $D = \{d_1, , d_N\}$ is a 2D displacement vector that added to the projected vertex locations. It is calculated as the difference between the groundtruth 2D feature locations and the projected vertex locations. To make it scale and rotation invariant, the displacement vector is normalized by a similarity transform that aligns the projected vertex locations to the mean shape. Thus, the unknown parameters for a frame can be summarized as:

$$\{f, u, v, m_{id}, H, D\}, \tag{4.7}$$

where f is the camera focal length, (u, v) is the optical center, m_{id} is the identity weight, and $H = \{R, T, m_{exp}\}$ represents the pose and expression weights for a given frame. Since the identity and camera intrinsic parameters (focal length and optical center) are

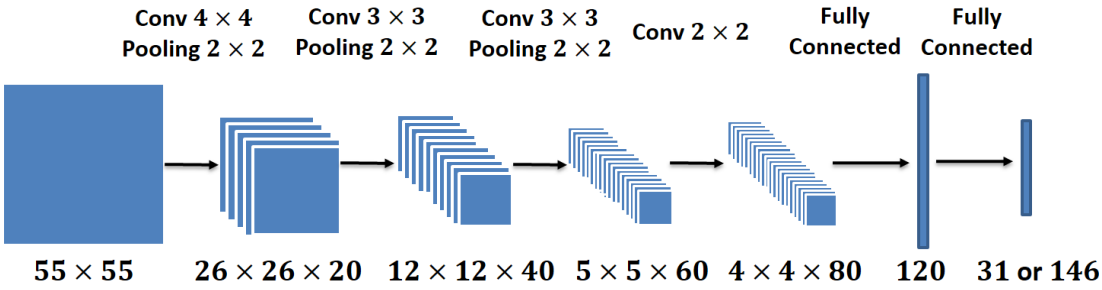


Figure 4.7: The net structure used in the CNN cascades. The output can be either a 31-dimentional pose and expression vector (3D Euler angle, 3D translation, and 25D expression weight) or a 146-dimentional 2D displacement vector (2D displacements for 73 facial feature points).

usually fixed for a video, we can pre-estimate them and assume they are known. Our task is then to infer the 3D head pose, expression H, and 2D displacements D from each video frame.

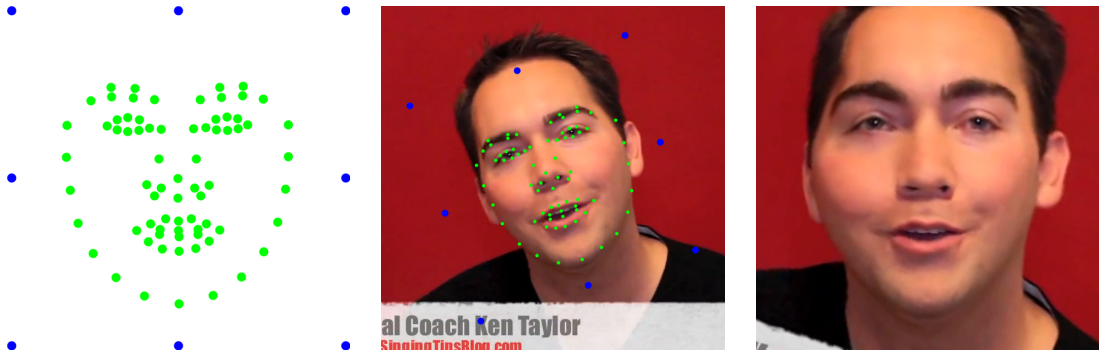
4.4.2 Cascaded Pose and Expression Regression Using CNN

We propose a CNN-based regression framework to infer pose, expression, and 2D feature locations from video frames. Given an input image and initial pose and expression parameters, we first extract a 2D texture in a canonical space, and feed it into the CNN. The CNN then outputs an increment for the pose and expression parameters. This process iterates a few times (3 in our framework) and the pose and expression parameters are progressively refined. At a final step, an additional CNN is used for regressing 2D displacements, which are added to the projected vertex locations to get the final 2D locations (Equation 4.8). The whole framework is summarized in Figure 4.6.

$$\begin{aligned} \Delta H^t &= \mathrm{CNN}_H^t(I, f, w_{id}; H^{t-1}), \\ D &= \mathrm{CNN}_D(I, f, w_{id}, H). \end{aligned} \tag{4.8}$$

The net structure used in the CNN cascades is shown in Figure 4.7. It takes an image

as input and outputs the increment of pose and expression parameters or the 2D displacements. The network consists of a set of convolutional layers, max-pooling (subsampling) layers, and fully connected layers. The rectified linear unit (ReLU) [78] is used as the activation function. The input image size is set to 39, 55, and 55 for the pose and expression regression, and 79 for the 2D displacements regression. We begin with a small image to let the net focus on global adjustment and then use larger images for detailed adjustments. We use Euclidean distance as the loss function, which minimizes the difference between net prediction and the shape increment ΔH or 2D displacement vector D.



(a) (b) (c)

Figure 4.8: Obtaining 2D texture at the canonical space. (a) the mean 2D shape (green) and augmented boundary points (blue); (b) the current projected 2D features (green) and the aligned boundary points (blue); (c) the obtained 2D texture using piecewise affine warping.

Canonical 2D texture space. One major challenge for CNN-based regression is how to effectively represent the accurateness of current estimate (pose and expression parameters or 2D displacements in our case) in the input format of the CNN: an image. The representation should have three properties. First, one should be able to tell from the image that what adjustments are needed such as a rotation change. Second, the representation should have internal correspondence for face shapes with different scales, locations, rota-

tions, and expressions, which ensures the features learned by CNN are robust and invariant to those changes. Finally, the representation should contain as complete facial pixels as possible, including both the internal face components and the contour region.

We propose a canonical 2D texture space for image representation. Specifically, we use a mean 2D shape of all training data S_{mean} as a reference shape. We further augment eight boundary points (Figure 4.8 (a)). Given an input image and current shape estimate, we first synthesize the 3D model and project corresponding vertices to image space to get a 2D shape using Equation 4.3. We then estimate a similarity transformation from the current shape S_{cur} to the mean shape:

$$\arg\min_{R,l,t} \sum_{k} \|s_{k,mean} - (lRs_{k,cur} + t)\|^2, \tag{4.9}$$

where R, l, t are the 2D rotation, scaling and translation respectively. The eight boundary points are then transferred to current image (Figure 4.8 (b)) using the inverse of the similarity transformation. Finally, we perform piecewise affine warping [5] to warp the current image I to the canonical space C (Figure 4.8 (c)). Briefly, we first apply Delaunay triangularization to the augmented meanshape. For every pixel x in the canonical 2D space, we first find the triangle it lies in and estimate affine transformation A from the meanshape to current shape for this triangle. We then obtain the pixel intensity through $C(x) = I(A(x))$. The warped image C is then used as the input to the CNN. This representation meets all three properties and ensures the effectiveness of CNN.

4.4.3 Results

We have collected 14897 images from labeled faces in the wild (LFW) [79], FaceWarehouse [49], and the Internet. We manually label and refine the 2D feature locations, and then use the reconstruction method described in Section 4.3.2 to build the 3D models. We randomly select 13408 images as the training images, and the remaining 1489 images are

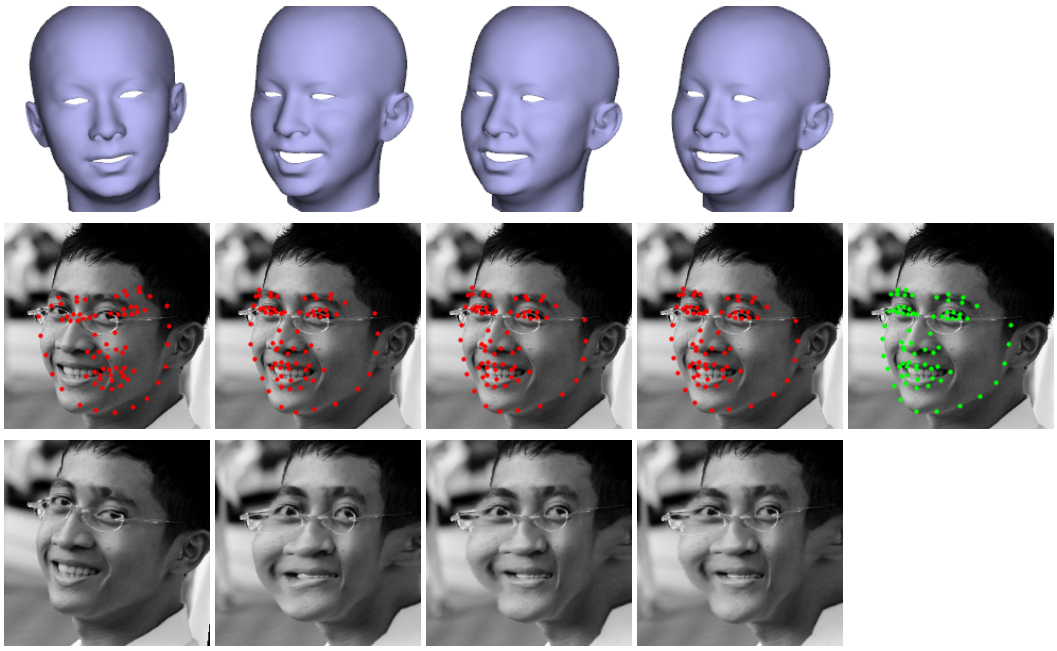


Figure 4.9: Visualization of the CNN-based pose and shape regression process. (a) the initial pose and expression, vertex projections, and the 2D texture; (b)-(d) the intermediate results of the pose and expression regression cascades; (e) the 2D feature locations with the regressed 2D displacements added.

used for testing. We then obtain the initializations for each image by replacing its pose and expression parameters with 25 randomly selected ones from other images. Thus, we have total 335k samples for training and 37k samples for testing.

Figure 4.9 visualizes the entire process of the CNN-based regression. Figure 4.10 shows some test results of the framework. As we can see, the proposed method produces accurate results and is robust to bad initializations. Figure 4.12 shows the 2D landmark error curves of the projected facial features after the pose and expression iterations and the final 2D locations after adding the 2D displacement vectors. The errors are normalized by the inter-ocular distance (*i.e.*, the distance between the two pupils). As shown in the figure, the 2D displacement regression effectively refines the 2D feature locations. The total time

Figure 4.10: Example test results of our framework. The first two rows show the initialization pose and expression and corresponding vertex projections. The third and fourth row show the pose and expression regression results and corresponding vertex projections. The fifth row shows the 2D landmarks after the final round of 2D displacements regression.

cost is 33 ms/frame based on our Matlab implementation which could be more efficient if implemented in C++ with GPU acceleration.

Finally, we have tested the proposed method on monocular videos and compared it with our method described in Section 4.3. Note that, for a fair comparison, the LBF-based regression model is also updated using the same training database. To apply the framework to video tracking, we first estimate the camera intrinsic parameters and identity weight from the first face frame using the method in Section 4.3. We then use the

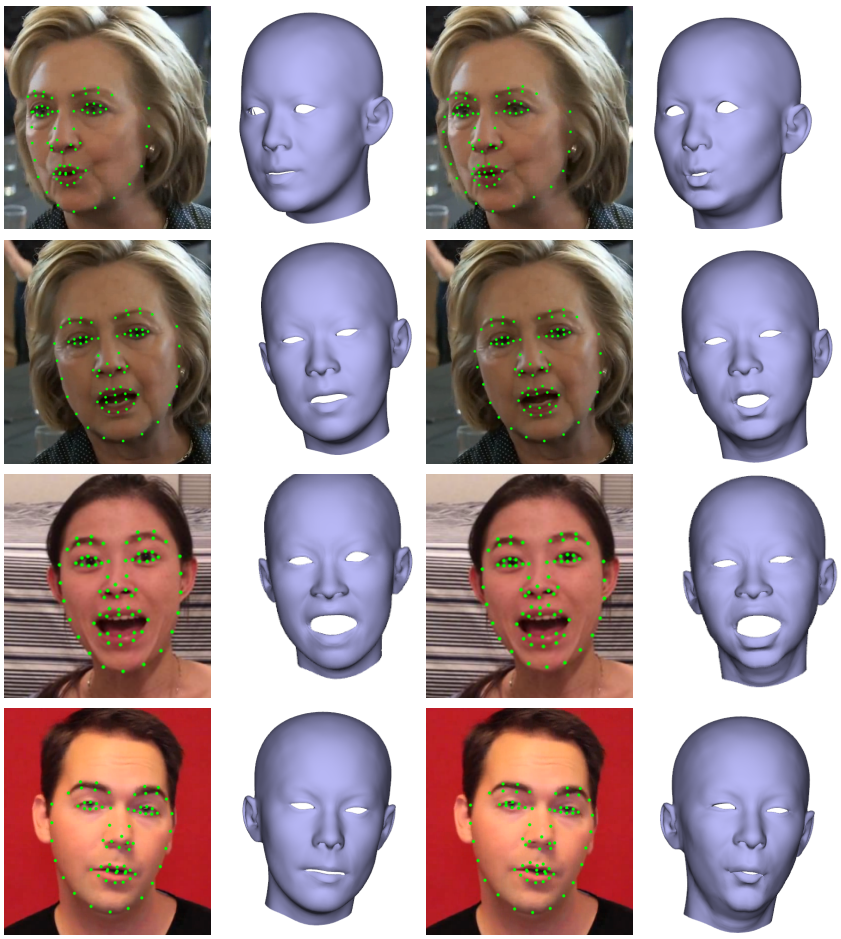


Figure 4.11: Video reconstruction results. (a)&(b) are results from the method described in Section 4.3, and (c)&(d) are obtained by the proposed CNN-based regression.

proposed framework to regress the pose and expression parameters as well as 2D feature locations for the rest of the frames. The pose is initialized using the previous result, and the expression weight is initialized using the nearest neighbor in the training database. The results (Figure 4.11) show that the CNN-based regression produces more expressive facial deformation, especially for the mouth region. The pose and expression fit the input face images better as the CNN cascades make full use of the image information instead of the sparse 2D features. On the other hand, the 2D landmark locations such as the contour

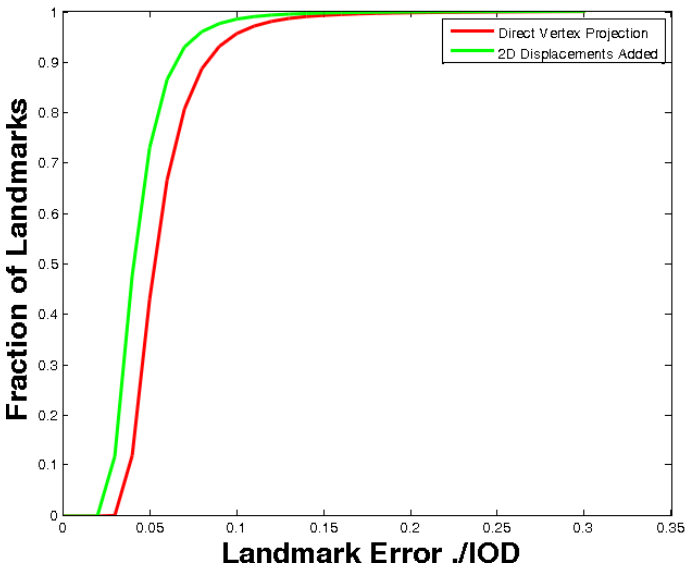


Figure 4.12: 2D landmark error curves measured on test data. The red curve shows the result after pose and expression iterations, and green curve shows the result after 2D displacements regression. The CNN for 2D displacements regression effectively improves the 2D landmark accuracy.

region could be further improved. We believe the 2D feature accuracy improvement can be achieved with deeper net structures which we will explore in the future.

4.5 Results and Applications

We have complemented the facial performance reconstruction component introduced in Section 4.3 with the proposed eye gaze tracker [80] and achieved realtime facial performance and 3D eye gaze capture. We demonstrate the power of our system on a large number of video sequences, including live video streams captured by a web camera and monocular video sequences downloaded from the Internet. Additionally, we have evaluated the effectiveness and accuracy of our system by comparing it against alternative methods. Table 4.1 reports computational time costs for each key component of our eye gaze tracker. Our system achieves realtime performance and runs at a frame rate of about 27 frames per second (fps). All our experiments were tested on a PC with Intel(R) Xeon(R)

CPU 3.3GHz, 16GB RAM, and NVIDIA GTX 780 graphics card.

Components	Timings(ms)
Expression reconstruction	23
Observation extraction	5
Eye gaze tracking	8.3
Failure detection	0.3
Total	36.6

Table 4.1: Computational cost of each key component of our system.

4.5.1 Test on Real Data

We have tested the effectiveness of our facial performance and eye gaze tracker on different subjects. In the online testing, we use a web camera with resolution 800 $\times$ 600. The result shows that the system can track the facial performance and eye gaze very robustly and accurately, even under fast eye gaze rolling, large head poses, facial expressions variations, and differences across individuals. Our system is also robust to large lighting changes as well as possible camera blur (Figure 4.13). In addition, we have shown that the system can successfully track the eye gaze and facial expressions for eight video clips, of which six are downloaded from the Internet and the rest are recorded by a common camera (Figure 4.14).

4.5.2 Comparison against Regression-based Method

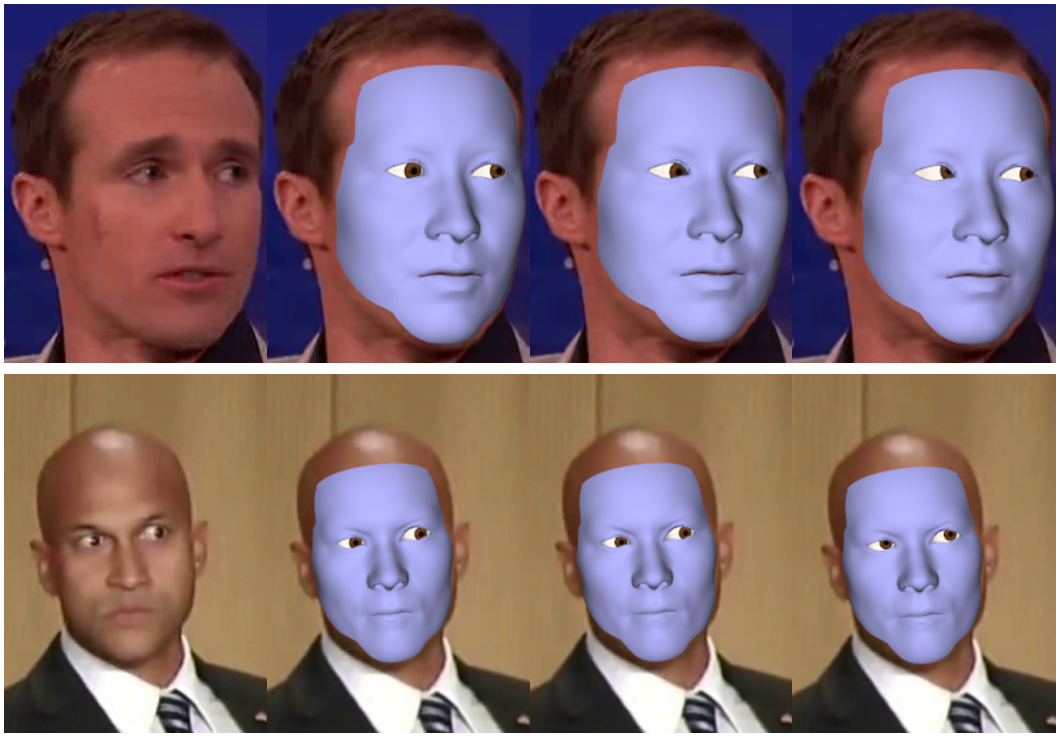
In this subsection, we evaluate the effectiveness and accuracy of our system by comparing against the state-of-the-art 2D facial feature detection/tracking system Face++ [81]. Face++ is a commercial facial landmark tracker developed by Megvii Technology Co.,Ltd. It is trained using the deep neural network model on a large set of labeled face images, and achieves highly accurate and robust performance for the facial landmark detection

Figure 4.13: Live demo of our system. Our system is robust and accurate even under significant lighting variations and extreme pose changes.

(a) (b) (c)

Figure 4.14: Our system is robust to pose and expression variations, difference across individuals, and partial occlusions. (a) the input video frame; (b) the tracked 2D facial features (green) and iris and pupil pixels (red); (c) the reconstruction results with 3D eye gaze.

tasks [82]. The company provides free API of facial landmark detection, which we used to test on a dataset to compare the accuracy on 2D pupil centers. Note that their method only focuses on the 2D facial landmark detection/tracking and only 2D pupil center is obtained, whereas our method can output 3D facial deformation and eye gaze motion.



(a) (b) (c) (d)

Figure 4.15: Comparison against Face++ landmark detector. (a) the input images; (b) the ground truth; (c) the eye gaze results of our method; (d) the eye gaze motion fitted by using the 2D pupil centers output by Face++ and our reconstructed 3D head pose and facial deformation.

The comparison is evaluated on eight video clips, which contain subjects of different races under various poses and lighting conditions (Figure 4.16). The image frames with closed eyes were excluded for evaluation because ground truth data for those frames are hard to obtain. We manually labeled the 2D pupil centers for the remaining image frames

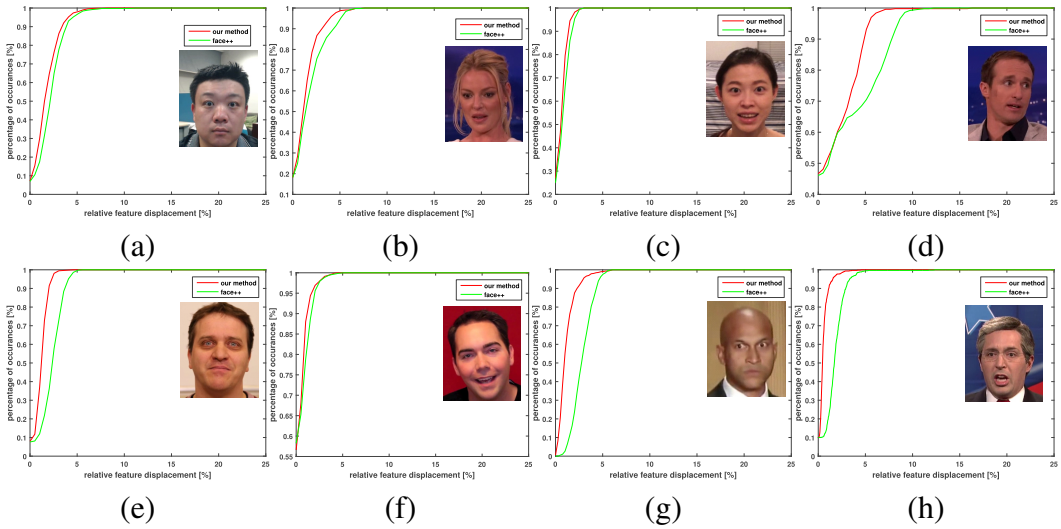


Figure 4.16: The quantitative evaluation of our algorithm and comparison against Face++. Note that only 2D pupil center locations are compared as Face++ does not reconstruct 3D eye gaze motion.

to get ground truth data for comparison. As suggested by BioID [83], the error metric is defined as follows:

$$E = \frac{max(\|p_l - {p'}_l\|, \|p_r - {p'}_r\|)}{\|{p'}_l - {p'}_r\|}, \tag{4.10}$$

where p_l and p_r are the centers of the left and right pupils obtained by the algorithm respectively. And ${p'}_l$ and ${p'}_r$ are corresponding ground truth data. Figure 4.16 shows that our method obtains steeper error distribution curve than Face++ in almost all the video clips, which demonstrates that our method is more robust and accurate than Face++.

Figure 4.15 shows some example results for comparisons. Note that Face++ does not track 3D eye gaze, head pose, or facial deformation. For visualization purpose, we back project the 2D pupil centers outputted by Face++ in 3D using the results we obtained in Section 4.3. The possible reason for our low error is that the eye gaze motion is loosely coupled with facial deformation, and tracking the eyeball motion sequentially after the

facial deformation reconstruction makes more sense than considering them as a whole.

4.5.3 Applications

With the tracked 3D head pose, facial deformation, and eye gaze, our system can easily retarget the 3D facial performance to a virtual avatar, capture eye gaze in realtime, and visualize the eye gaze with edited iris and pupil texture to generate appealing special effects at the iris and pupil region.

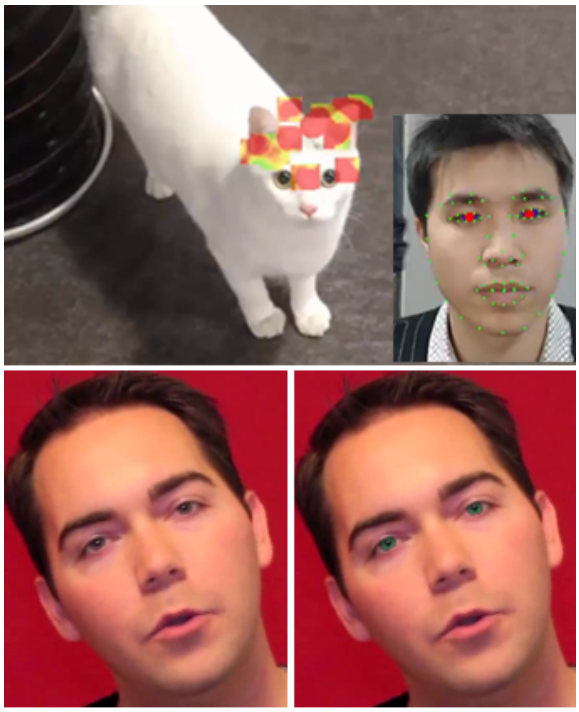


Figure 4.17: Realtime eye gaze capture (top row) and eye gaze visualization (bottom row). Top row: the user is asked to watch a video in front of a camera and the focus locations on screen space are visualized using heatmap; bottom row: since we calibrated the eyeball center and the size of the iris and pupil region, we can edit the iris and pupil appearance by applying a new iris and pupil texture, projecting it to image plane and blending it with the original image.

Facial and eye gaze retargeting. Given the facial expression parameters as well as the

eye gaze states of both eyes, we can easily retarget the tracked motion to a virtual avatar. Specifically, we first fit the identity parameters and calibrate the eyeball center and the size of iris and pupil region for a given avatar. Note that the eyeball information can also be provided by the artist. Next, with the known identity parameters and eyeball model, we can directly transfer the expression parameters, pose as well as the pupil centers (ϕ, θ) from the source, and drive the avatar to perform similar expression, pose, and eye gaze motion at runtime.

Realtime eye gaze capture. With the reconstructed head orientations and tracked gaze states, we can locate the eye gaze points (fixations) in screen space at runtime and visualize them using the heatmap. The eye gaze analysis is widely used in the study of human psychology and sociology. To start the process, we calibrate a linear transformation between the camera image plane and the screen plane by asking the user to focus on some pre-defined known screen locations (*e.g.*, left top, middle top, right top, etc.) and solving a least-squares problem. After calibration, we can map the eye gaze points from the image plane to the screen space. We then assign gaussian weights to the corresponding screen region. We also set a timer for each pixel to cool down the old focus locations. Figure 4.5.3 shows the realtime eye gaze capture using the heatmap in which the user is tracking a running cat.

Eye gaze visualization. One unique feature of our system is that it not only captures eye gaze motion but also calibrates the eyeball center and the size of iris and pupil region. Thus, we are able to obtain the exact iris and pupil regions in each video frame by projecting the eyeball onto the image plane using a binary texture. In this application, the iris and pupil appearance is edited by applying a new iris and pupil texture, projecting it to the image plane, and blending the projected texture with the original image (Figure 4.5.3). Note that this process is fully automatic and no manual refinement is needed. Though the performance has large pose and eye gaze variations as well as frequent eye blinkings, the

results are continuous and natural, which clearly demonstrates the accuracy and robustness of our system.

4.6 Discussion

In this section, we demonstrate an end-to-end realtime system that captures the coordinated movements of 3D head poses, facial expression deformation, and eye gaze using a monocular video camera. We improve the efficiency and accuracy of 3D facial tracker by introducing a local binary feature based 2D landmark regression and a novel convolutional neural network based pose and expression regression, and enhance it by adding a 3D eye gaze tracker that automatically and robustly tracks the 3D eye gaze in the Maximum A Posterior (MAP) framework. Our system is appealing for facial and eye gaze capture because it is fully automatic, runs in realtime, and offers the lowest cost and a simplified setup. We have tested our system on both live video streams and Internet videos demonstrating its accuracy and robustness under a variety of uncontrolled lighting conditions and overcoming significant differences of races, genders, shapes, poses, and expressions across individuals.

The current system has a few limitations. First, the eye gaze tracking could fail to detect the 2D pupil center locations when the subject performs some peculiar eye gaze motion such as rolling white eyes. Note that the system could still output visually natural eye gaze motion with the failure detection handling component. Additionally, as a video-based technique, our system will fail to track the facial performance and eye gaze if many of the facial features cannot be observed in the video frames. Finally, the reconstructed 3D facial mesh does not contain fine geometric details. If required, one could combine our eye gaze tracking algorithm with the techniques proposed by Cao et al. [59] for realtime high-fidelity facial and eye gaze animation.

The CNN-based pose and expression regression proposed in Section 4.4 is an appeal-

ing alternative for facial performance capture using monocular videos. It takes the entire face image instead of sparse 2D facial features as input and obtains 3D head pose and expression as well as 2D facial features in a single step. Experiments show the proposed method is accurate and robust to bad initializations. Comparing with our two-step reconstruction (Section 4.3.2), it produces more expressive facial deformation, especially for the mouth region. In the future, we will explore more net structures for a better accuracy and combine the proposed method with our eye gaze tracking framework.

We will also add more training images to obtain a more robust iris and pupil classifier. Moreover, we are particularly interested in using our captured facial and eye gaze data for further study. For example, we can combine the captured 3D facial deformation and eye gaze with the audio signal in the video, and study the relation between them similar to Liu et al. [84]. We can then build the statistical model to leverage visual and audio signals for a more natural and robust facial and eye gaze performance capture.

5. CONCLUSIONS AND FUTURE WORK

Capturing high-quality facial performance for common users using a single video camera is challenging but will have broad impacts for many fields such as movies, games, human computer interaction, and virtual/augmented reality. In this dissertation, we have introduced a set of novel algorithms and systems for video-based facial performance capture and animation, covering different aspects of the output such as 2D facial features, 3D large-scale facial deformation, and fine-scale facial details. A realtime facial performance reconstruction/tracking module is presented and complemented with an efficient 3D eye gaze tracker to achieve realtime 3D eye gaze tracking.

In Section 2, we propose an accurate and robust facial feature detection/tracking algorithm that quickly and accurately locates facial features from video frames and applied this method to RGBD images obtained by a Kinect camera. The key idea is to combine the advantages of local detection, AAMs, and temporal coherence. At one end, local feature detectors can robustly infer the locations of facial features in single images but often with less accurate and unstable results. At the other end, facial registration using AAMs can achieve sub-pixel accuracy but is often sensitive to the initialization due to the gradient descent optimization. We combine both techniques into a Lucas-Kanade registration framework to enjoy benefits at both ends. We further improve the robustness and accuracy by incorporating the temporal coherence. One limitation is the gradient based optimization leaves little time budget for advanced applications based on it.

Section 3 describes an end-to-end system that captures high-fidelity facial performances from monocular videos. The key idea is to utilize the detected facial features and per-pixel shading cues, along with multilinear facial models, to infer 3D head poses, large-scale facial deformation, and fine-scale facial detail across the entire sequence. One challenge

for this work is the lack of discernible features on most facial regions such as cheeks. It is possible that the reconstructed large-scale deformation based on the sparse tracked 2D facial features does not fit the real facial geometry well. We propose to iterate the reconstruction process on large-scale facial deformation and fine-scale facial details to further use the per-pixel shading cues to refine the facial geometry. The final system is robust and fully automatic and captures both large-scale facial deformation and fine-scale facial details from monocular videos. However, it requires extracting key frames from the entire video and cannot be applied to realtime applications.

In Section 4, we overcome the two limitations mentioned above and present a realtime facial performance capture system that simultaneously captures 3D head poses, large-scale facial deformation, and 3D eye gaze using a single RGB camera. We first refine the accuracy and robustness of the 2D feature tracking and 3D large-scale facial reconstruction component and improve its efficiency to 43 fps. We then present a convolutional neural network based pose and shape regression framework which is an appealing alternative to this component. Finally, we complement the component with an efficient 3D eye gaze tracker that utilizes both per-frame observation and temporal coherence and achieve 3D eye gaze animation in realtime. The system is applicable to common users that have a video camera and has direct applications such as realtime facial animation, screen-space eye gaze capture, and video editing. It also has the potential to be used to track facial performance and eye gaze for head-mounted VR display.

Though the aforementioned systems have achieved our goal mentioned in the beginning of this dissertation, which is to reconstruct 3D facial performances from monocular videos (*e.g.*, Internet videos) automatically and efficiently, there are still many fields worth more exploration. First, the expression ability of the 3D shape is limited to the database we use for building the multilinear model. The current database [49] contains 150 identities and the majority are Asian descent. Increasing the database to include more variety of

identities and expressions will improve the reconstruction accuracy.

Another immediate future work is to refine the CNN-based pose and expression regression method proposed in Section 4 and integrate it with our eye gaze tracking framework. We can also refine the 3D training data with manual adjustments and explore and evaluate more net structures for better regression accuracy. It is also worth exploring using CNN to estimate 3D eye gaze from the input eye image. The whole system is expected to deliver better facial deformation and more accurate 3D eye gaze and is more efficient.

Our facial performance capture system can be further extended to capture facial and eye movements for people wearing a head-mounted display. The key challenge here is only partial face is visible. One way to address it is to use the autoencoder neural network to recover the image of the occluded facial region. Additional camera inside the display could also be used to capture eye movement. Another possible solution is to use CNN to infer the entire facial expression directly from the observed mouth region. The result can be combined into our facial capture system as a prior.

In Section 3 and Section 4, our target scenario is uncontrolled Internet videos. However, the recovered identity and corresponding expression basis could be inaccurate because the videos might not cover all possible facial deformations of the user. Thus, another future direction is to reconstruct person-specific high-fidelity blendshapes from a video sequence that is specially taken with representative facial expressions or a large enough photo gallery of the subject. We first recover large-scale facial geometry from the video/image gallery. The shading information can then be used to refine the large-scale geometry similar to the process in Section 3.6. Additionally, we can add deformation gradients to represent the out-of-space model adjustments. The reconstructed high-fidelity blendshapes can then be used for facial animation and video editing. This is an important problem because it has the potential to compete with the process of acquiring person-specific blendshapes in the movie industry, which usually requires the actor wearing mark-

ers sitting still in front of special equipment such as a large camera array.

REFERENCES

[1] I. Kemelmacher-Shlizerman and R. Basri, "3d face reconstruction from a single image using a single reference face shape," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 33, no. 2, pp. 394–405, 2011.

[2] P. Garrido, L. Valgaerts, C. Wu, and C. Theobalt, "Reconstructing detailed dynamic face geometry from monocular video," *ACM Transactions on Graphics*, vol. 32, no. 6, p. 158, 2013.

[3] S. Ren, X. Cao, Y. Wei, and J. Sun, "Face alignment at 3000 fps via regressing local binary features," in *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1685–1692, 2014.

[4] T. F. Cootes, G. J. Edwards, and C. J. Taylor, "Active appearance models," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 23, no. 6, pp. 681–685, 2001.

[5] S. Baker and I. Matthews, "Lucas-kanade 20 years on: A unifying framework," *International Journal of Computer Vision*, vol. 56, no. 3, pp. 221–255, 2004.

[6] J. M. Saragih, S. Lucey, and J. F. Cohn, "Deformable model fitting by regularized landmark mean-shift," *International Journal of Computer Vision*, vol. 91, no. 2, pp. 200–215, 2011.

[7] T. Baltrusaitis, P. Robinson, and L. Morency, "3d constrained local model for rigid and non-rigid facial tracking," in *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2610–2617, 2012.

[8] Microsoft, "Kinect for windows sdk." http://msdn.microsoft.com/en-us/library/jj130970.aspx/, 2013.

[9] D. Cristinacce and T. Cootes, "Feature detection and tracking with constrained local models," in *British Machine Vision Conference*, vol. 3, pp. 929–938, 2006.

[10] Y. Wang, S. Lucey, and J. F. Cohn, "Enforcing convexity for improved alignment with constrained local models," in *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1–8, 2008.

[11] D. Vukadinovic and M. Pantic, "Fully automatic facial feature point detection using gabor feature based boosted classifiers," in *IEEE International Conference on Systems, Man and Cybernetics*, vol. 2, pp. 1692–1698, 2005.

[12] M. Dantone, J. Gall, G. Fanelli, and L. Van Gool, "Real-time facial feature detection using conditional regression forests," in *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2578–2585, IEEE, 2012.

[13] M. Valstar, B. Martinez, X. Binefa, and M. Pantic, "Facial point detection using boosted regression and graph models," in *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2729–2736, 2010.

[14] X. Zhu and D. Ramanan, "Face detection, pose estimation, and landmark localization in the wild," in *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2879–2886, 2012.

[15] P. N. Belhumeur, D. W. Jacobs, D. Kriegman, and N. Kumar, "Localizing parts of faces using a consensus of exemplars," in *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 545–552, 2011.

[16] T. F. Cootes, M. C. Ionita, C. Lindner, and P. Sauer, "Robust and accurate shape model fitting using random forest regression voting," in *European Conference on Computer Vision*, pp. 278–291, 2012.

[17] M. Zhou, L. Liang, J. Sun, and Y. Wang, "Aam based face tracking with temporal matching and face segmentation," in *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 701–708, 2010.

[18] Y. Amit and D. Geman, "Shape quantization and recognition with randomized trees," *Neural Computation*, vol. 9, no. 7, pp. 1545–1588, 1997.

[19] V. Lepetit and P. Fua, "Keypoint recognition using randomized trees," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 28, no. 9, pp. 1465–1479, 2006.

[20] J. Shotton, A. Fitzgibbon, M. Cook, T. Sharp, M. Finocchio, R. Moore, A. Kipman, and A. Blake, "Real-time human pose recognition in parts from a single depth image," in *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1297–1304, 2011.

[21] D. Comaniciu and P. Meer, "Mean shift: A robust approach toward feature space analysis," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 5, pp. 603–619, 2002.

[22] Y. Lamdan and H. Wolfson, "Geometric hashing: A general and efficient model-based recognition scheme," in *IEEE International Conference on Computer Vision*, pp. 238–249, 1988.

[23] I. Matthews and S. Baker, "Active appearance models revisited," *International Journal of Computer Vision.*, vol. 60, no. 2, pp. 135–164, 2004.

[24] D. A. Ross, J. Lim, R.-S. Lin, and M.-H. Yang, "Incremental learning for robust visual tracking," *International Journal of Computer Vision*, vol. 77, no. 1-3, pp. 125–141, 2008.

[25] G. Fanelli, M. Dantone, J. Gall, A. Fossati, and L. Van Gool, "Random forests for real time 3d face analysis," *International Journal of Computer Vision*, vol. 101, no. 3, pp. 437–458, 2013.

[26] X. Cao, Y. Wei, F. Wen, and J. Sun, "Face alignment by explicit shape regression," in *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2887–2894, 2012.

[27] B. Guenter, C. Grimm, D. Wood, H. Malvar, and F. Pighin, "Making Faces," in *ACM SIGGRAPH*, pp. 55–66, 1998.

[28] B. Bickel, M. Botsch, R. Angst, W. Matusik, M. Otaduy, H. Pfister, and M. Gross, "Multi-scale capture of facial geometry and motion," *ACM Transactions on Graphics*, vol. 26, no. 3, pp. 33:1–33:10, 2007.

[29] H. Huang, J. Chai, X. Tong, and H.-T. Wu, "Leveraging motion capture and 3d scanning for high-fidelity facial performance acquisition," *ACM Transactions on Graphics*, vol. 30, no. 4, pp. 74:1–74:10, 2011.

[30] L. Zhang, N. Snavely, B. Curless, and S. Seitz, "Spacetime faces: high resolution capture for modeling and animation," *ACM Transactions on Graphics*, vol. 23, no. 3, pp. 548–558, 2004.

[31] W.-C. Ma, A. Jones, J.-Y. Chiang, T. Hawkins, S. Frederiksen, P. Peers, M. Vukovic, M. Ouhyoung, and P. Debevec, "Facial performance synthesis using deformation-driven polynomial displacement maps," *ACM Transactions on Graphics*, vol. 27, no. 5, pp. 121:1–121:10, 2008.

[32] H. Li, B. Adams, L. J. Guibas, and M. Pauly, "Robust single-view geometry and motion reconstruction," *ACM Transactions on Graphics*, vol. 28, no. 5, pp. 175:1–175:10, 2009.

[33] T. Weise, H. Li, L. Van Gool, and M. Pauly, "Face/off: live facial puppetry," in *ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pp. 7–16, 2009.

[34] D. Bradley, W. Heidrich, T. Popa, and A. Sheffer, "High resolution passive facial performance capture," *ACM Transactions on Graphics*, vol. 29, no. 4, pp. 41:1–41:10, 2010.

[35] T. Beeler, B. Bickel, P. Beardsley, B. Sumner, and M. Gross, "High-quality single-shot capture of facial geometry," *ACM Transactions on Graphics*, vol. 29, no. 4, pp. 40:1–40:9, 2010.

[36] T. Beeler, F. Hahn, D. Bradley, B. Bickel, P. Beardsley, C. Gotsman, R. W. Sumner, and M. Gross, "High-quality passive facial performance capture using anchor frames," *ACM Transactions on Graphics*, vol. 30, no. 4, pp. 75:1–75:10, 2011.

[37] L. Valgaerts, C. Wu, A. Bruhn, H.-P. Seidel, and C. Theobalt, "Lightweight binocular facial performance capture under uncontrolled lighting," *ACM Transactions on Graphics*, vol. 31, no. 6, pp. 187:1–187:11, 2012.

[38] V. Blanz, C. Basso, T. Poggio, and T. Vetter, "Reanimating faces in images and video," *Computer Graphics Forum*, vol. 22, no. 3, pp. 641–650, 2003.

[39] D. Vlasic, M. Brand, H. Pfister, and J. Popović, "Face transfer with multilinear models," *ACM Transactions on Graphics*, vol. 24, no. 3, pp. 426–433, 2005.

[40] C. Cao, Y. Weng, S. Lin, and K. Zhou, "3d shape regression for real-time facial animation," *ACM Transactions on Graphics*, vol. 32, no. 4, pp. 41:1–41:10, 2013.

[41] S. Suwajanakorn, I. Kemelmacher-Shlizerman, and S. M. Seitz, "Total moving face reconstruction," in *European Conference on Computer Vision*, pp. 796–812, 2014.

[42] P. Garrido, L. Valgaert, C. Wu, and C. Theobalt, "Reconstructing detailed dynamic face geometry from monocular video," *ACM Transactions on Graphics*, vol. 32, no. 6, pp. 158:1–158:10, 2013.

[43] C. Wu, K. Varanasi, Y. Liu, H.-P. Seidel, and C. Theobalt, "Shading-based dynamic shape refinement from multi-view video under general illumination," in *International Conference on Computer Vision*, pp. 1108–1115, 2011.

[44] T. Weise, S. Bouaziz, H. Li, and M. Pauly, "Realtime performance-based facial animation," *ACM Transactions on Graphics*, vol. 30, no. 4, pp. 77:1–77:10, 2011.

[45] S. Bouaziz, Y. Wang, and M. Pauly, "Online modeling for realtime facial animation," *ACM Transactions on Graphics*, vol. 32, no. 4, pp. 40:1–40:10, 2013.

[46] H. Li, J. Yu, Y. Ye, and C. Bregler, "Realtime facial animation with on-the-fly correctives," *ACM Transactions on Graphics*, vol. 32, no. 4, pp. 42:1–42:10, 2013.

[47] P.-L. Hsieh, C. Ma, J. Yu, and H. Li, "Unconstrained realtime facial performance capture," in *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1675–1683, 2015.

[48] Y. Dai, H. Li, and M. He, "A simple prior-free method for non-rigid structure-from-motion factorization," in *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2018–2025, 2012.

[49] C. Cao, Y. Weng, S. Zhou, Y. Tong, and K. Zhou, "Facewarehouse: a 3d facial expression database for visual computing," *IEEE Transations on Visualizationa and Computer Graphics*, vol. 20, no. 3, pp. 413–425, 2014.

[50] P. Å. Wedin, "On angles between subspaces of a finite dimensional inner product space," in *Matrix Pencils*, pp. 263–285, 1983.

[51] B. Horn and M. Brooks, *Shape from Shading*. MIT Press: Cambridge, MA, 1989.

[52] R. Basri and D. W. Jacobs, "Lambertian reflectance and linear subspaces," *IEEE Transactions on Pattern Analysis and Machine Intelligence.*, vol. 25, no. 2, pp. 218–233, 2003.

[53] B. K. Horn and M. J. Brooks, "The variational approach to shape from shading," *Computer Vision, Graphics, and Image Processing*, vol. 33, no. 2, pp. 174–208, 1986.

[54] M. I. A. Lourakis, "levmar: Levenberg-Marquardt nonlinear least squares algorithms in C/C++." http://users.ics.forth.gr/ lourakis/levmar/, 2009.

[55] P. Pérez, M. Gangnet, and A. Blake, "Poisson image editing," *ACM Transactions on Graphics*, vol. 22, no. 3, pp. 313–318, 2003.

[56] M. Bunnell, "Adaptive tessellation of subdivision surfaces with displacement mapping," in *GPU Gems 2*, pp. 109–122, Addison-Wesley: Boston, MA, 2005.

[57] R. W. Sumner and J. Popović, "Deformation transfer for triangle meshes," *ACM Transactions on Graphics*, vol. 23, no. 3, pp. 399–405, 2004.

[58] C. Cao, Q. Hou, and K. Zhou, "Displaced dynamic expression regression for real-time facial tracking and animation," *ACM Transactions on Graphics*, vol. 33, no. 4, p. 43, 2014.

[59] C. Cao, D. Bradley, K. Zhou, and T. Beeler, "Real-time high-fidelity facial performance capture," *ACM Transactions on Graphics*, vol. 34, no. 4, p. 46, 2015.

[60] S. Saito, T. Li, and H. Li, "Real-time facial segmentation and performance capture from rgb input," *arXiv preprint arXiv:1604.02647*, 2016.

[61] H. Li, L. Trutoiu, K. Olszewski, L. Wei, T. Trutna, P.-L. Hsieh, A. Nicholls, and C. Ma, "Facial performance sensing head-mounted display," *ACM Transactions on Graphics*, vol. 34, no. 4, p. 47, 2015.

[62] J. M. Saragih, S. Lucey, and J. F. Cohn, "Real-time avatar animation from a single image," in *IEEE International Conference on Automatic Face & Gesture Recognition and Workshops*, pp. 117–124, 2011.

[63] T. Baltrušaitis, P. Robinson, and L.-P. Morency, "3d constrained local model for rigid and non-rigid facial tracking," in *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2610–2617, 2012.

[64] X. Xiong and F. De la Torre, "Supervised descent method and its applications to face alignment," in *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 532–539, 2013.

[65] Y. Sun, X. Wang, and X. Tang, "Deep convolutional network cascade for facial point detection," in *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3476–3483, 2013.

[66] H. Fan and E. Zhou, "Approaching human level facial landmark localization by deep learning," *Image and Vision Computing*, vol. 47, pp. 27–35, 2016.

[67] A. Jourabloo and X. Liu, "Large-pose face alignment via cnn-based dense 3d model fitting," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

[68] X. Zhu, Z. Lei, X. Liu, H. Shi, and S. Z. Li, "Face alignment across large poses: A 3d solution," *arXiv preprint arXiv:1511.07212*, 2015.

[69] C. H. Morimoto and M. Flickner, "Real-time multiple face detection using active illumination," in *IEEE International Conference on Automatic Face and Gesture Recognition*, pp. 8–13, 2000.

[70] P. M. Corcoran, F. Nanu, S. Petrescu, and P. Bigioi, "Real-time eye gaze tracking for gaming design and consumer electronics systems," *IEEE Transactions on Consumer Electronics*, vol. 58, no. 2, pp. 347–355, 2012.

[71] J. Huang and H. Wechsler, "Eye detection using optimal wavelet packets and radial basis functions (rbfs)," *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 13, no. 7, pp. 1009–1025, 1999.

[72] W. Huang and R. Mariani, "Face detection and precise eyes location," in *IEEE Conference on International Conference on Pattern Recognition*, vol. 4, pp. 722–727, 2000.

[73] S. Kawato and J. Ohya, "Real-time detection of nodding and head-shaking by directly detecting and tracking the between-eyes," in *International Conference on Automatic Face and Gesture Recognition*, pp. 40–45, 2000.

[74] Y.-l. Tian, T. Kanade, and J. F. Cohn, "Dual-state parametric eye tracking," in *IEEE International Conference on Automatic Face and Gesture Recognition*, pp. 110–115, 2000.

[75] D. Vlasic, M. Brand, H. Pfister, and J. Popović, "Face transfer with multilinear models," *ACM Transactions on Graphics*, vol. 24, no. 3, pp. 426–433, 2005.

[76] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems*, pp. 1097–1105, 2012.

[77] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *arXiv preprint arXiv:1512.03385*, 2015.

[78] X. Glorot, A. Bordes, and Y. Bengio, "Deep sparse rectifier neural networks," in *International Conference on Artificial Intelligence and Statistics*, vol. 15, p. 275, 2011.

[79] G. B. Huang, M. Ramesh, T. Berg, and E. Learned-Miller, "Labeled faces in the wild: A database for studying face recognition in unconstrained environments," tech. rep.,

University of Massachusetts, Amherst, 2007.

[80] C. Wang, F. Shi, S. Xia, and J. Chai, "Realtime 3d eye gaze animation using a single rgb camera," *ACM Transactions on Graphics*, vol. 35, no. 4, p. 118, 2016.

[81] Megvii Technology, "Face++ sdk." http://www.faceplusplus.com.cn, 2015.

[82] C. Sagonas, G. Tzimiropoulos, S. Zafeiriou, and M. Pantic, "300 faces in-the-wild challenge: The first facial landmark localization challenge," in *IEEE International Conference on Computer Vision Workshops*, pp. 397–403, 2013.

[83] BioID, "Bioid face database." https://www.bioid.com/About/BioID-Face-Database, 2015.

[84] Y. Liu, F. Xu, J. Chai, X. Tong, L. Wang, and Q. Huo, "Video-audio driven real-time facial animation," *ACM Transactions on Graphics*, vol. 34, no. 6, p. 182, 2015.